

CDNI
Internet-Draft
Intended status: Informational
Expires: August 27, 2012

R. van Brandenburg
O. van Deventer
TNO
February 24, 2012

Models for adaptive-streaming-aware CDN Interconnection
draft-brandenburg-cdni-has-00

Abstract

This document presents thoughts on the potential impact of supporting HTTP Adaptive Streaming technologies in CDN Interconnection scenarios. Our intent is to spur discussion on how the different CDNI interfaces should deal with content delivered using adaptive streaming technologies.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 27, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Terminology	3
2. HTTP Adaptive Streaming aspects relevant to CDNI	4
2.1. Segmentation versus Fragmentation	4
2.2. Addressing chunks	5
2.2.1. Full Locator	6
2.2.2. Relative Locator	7
2.2.3. Chunk Request Routing	7
3. Impact of HAS on CDNI	8
3.1. General	8
3.1.1. On the definition of a Content Item in CDNI	8
3.1.2. General CDNI-HAS Requirements	10
3.2. Impact on Request Routing Interface	10
3.2.1. Dealing with manifest files	10
3.2.2. HAS Request Routing	11
3.2.3. Dividing content over multiple nodes	12
3.2.4. HAS Requirements for the CDNI Request Routing Interface	12
3.3. Impact on Metadata Interface	12
3.3.1. HAS-specific Metadata	12
3.3.2. HAS Requirements for the CDNI Metadata Interface	13
3.4. Impact on Logging Interface	13
3.4.1. Log processing	13
3.4.2. HAS Requirements for the CDNI Logging Interface	14
3.5. Impact on Control Interface	14
3.5.1. HAS Requirements for the CDNI Control Interface	14
4. IANA Considerations	14
5. Security Considerations	14
6. References	14
6.1. Normative References	14
6.2. Informative References	15
Authors' Addresses	15

1. Introduction

HTTP Adaptive Streaming (HAS) is an umbrella term for various HTTP-based streaming technologies that allow a client to adaptively switch between multiple bitrates depending on current network conditions. A defining aspect of HAS is that, since it is based on HTTP, it is a session-less pull-based mechanism, with a client actively requesting content segments, instead of the content being pushed to the client by a server. Due to this session-less nature, media servers delivering content using HAS often show different characteristics when compared with media servers delivering content using traditional streaming methods such as RTP/RTSP, RTMP and MMS. This document presents a discussion on what the impact of these different characteristics is to the CDNI interfaces. The scope of this document is explicitly not to define solutions, but merely to identify different methods of handling HAS in a CDN and the potential problems when using HAS in a CDNI context. The issues identified in this document may be used as input for defining HAS-specific requirements for the CDNI interfaces.

1.1. Terminology

This document uses the terminology defined in [I-D.ietf-cdni-problem-statement].

In addition, the following terms are used throughout this document:

Content Item: A uniquely adressable content element in a CDN. A content item is defined by the fact that it has its own Content Metadata associated with it. It is the object of a request routing operation in a CDN. An example of a Content Item is a video file/stream, an audio file/stream or an image file. For a discussion on what may constitute a Content Item with regards to HAS, see section 3.1.1.

Chunk: A fixed length element that is the result of a segmentation or fragmentation operation being performed on a single encoding of the Original Content. A chunk is independently, and uniquely, addressable. Depending on the way a Chunk is stored, it may also be referred to as a Segment or Fragment.

Fragment: A specific form of chunk (see section 2.1). A fragment is stored as part of a larger file that includes all chunks that are part of the Chunk Collection.

Segment: A specific form of chunk (see section 2.1). A segment is stored as a single file from a file system perspective.

Original Content: Unchunked content that is the basis for a segmentation of fragmentation operation. Based on Original Content, multiple alternative encodings, resolutions or bitrates may be derived, each of which may be fragmented or segmented.

Chunk Collection: The set of all chunks that are the result of a single segmentation or fragmentation operation being performed on a single encoding of the Original Content. A Chunk Collection is described in a manifest file.

Content Collection: The set of all Chunk Collections that are derived from the same Original Content. A Content Collection may consist of multiple Chunk Collections, each being a single encoding, or variant, of the Original Content. A Content Collection may be described by one or more manifest files.

Manifest File: A manifest file, also referred to as Media Presentation Description (MPD) file, is a file that list the way the content has been chunked and where the various chunks are located (in the case of segments) or how they can be addressed (in the case of fragments).

2. HTTP Adaptive Streaming aspects relevant to CDNI

In the last couple of years, a wide variety of HAS-like protocols have emerged. Among them are proprietary solutions such as Apple's HTTP Live Streaming (HLS), Microsoft's Smooth Streaming (MSS) and Adobe's HTTP Dynamic Streaming (HDS), and various standardized solutions such as 3GPP AHS (AHS) and MPEG DASH (DASH). While all of these technologies share a common set of features, each has its own defining elements. This chapter will look at some of the differences between these technologies and how these differences might be relevant to CDNI. In particular, section 2.1 will describe the various methods to store HAS content and section 2.2 will list three methods that are used to address HAS content in a CDN.

2.1. Segmentation versus Fragmentation

All HAS implementations are based around a concept referred to as chunking: the concept of having a server split content up in numerous fixed length chunks, which are independently decodable. By sequentially requesting and receiving chunks, a client can recreate and play out the content. An advantage of this mechanism is that it allows a client to seamlessly switch between different encodings of the same Original Content. Before requesting a particular chunk, a client can choose between multiple alternatives of the same chunk, irrespective of the encoding of the chunks it has requested earlier.

NOTE: The set of all chunks belonging to a single encoding, and thus the result of a single chunking operation, will from now on be referred to as a Chunk Collection. The set of all encodings of the same Original Content will be referred to as a Content Collection. A Content Collection can therefore consist of multiple Chunk Collections.

While every HAS implementation uses some form of chunking, not all implementations store the resulting chunks in the same way. In general, there are two distinct methods of performing chunking and storing the results: segmentation and fragmentation.

- With segmentation, which is for example mandatory in all versions of HLS prior to version 7, the chunks, in this case also referred to as segments, are stored completely independent from each other, with each segment being stored as a separate file from a file system perspective. This means that each segment has its own unique URL with which it can be retrieved.
- With fragmentation (or virtual segmentation), which is for example used in Microsoft's Smooth Streaming, all chunks, or fragments, belonging to the same Chunk Collection are stored together, as part of a single file. While there are a number of container formats which allow for storing this type chunked content, Fragmented MP4 is most commonly used. With fragmentation, a specific chunk is addressable by subfixing the common file URL with an identifier uniquely identifying the chunk one is interested in, either by timestamp, by byterange, or in some other way.

While one can argue about the merits of each of these two different methods of handling chunks, both have their advantages and drawbacks in a CDN environment. For example, fragmentation is often regarded as a method that introduces less overhead, both from a storage and processing perspective. Segmentation on the other hand, is regarded as being more flexible and efficient with regards to caching. In practice, current HAS implementations increasingly support both methods.

2.2. Addressing chunks

In order for a client to request chunks, either in the form of segments or in the form of fragments, it needs to know how the content has been chunked and where to find the chunks. For this purpose, most HAS protocols use a concept that is often referred to as a manifest file (also known as Media Presentation Description, or MPD): a file that lists the way the content has been chunked and where the various chunks are located (in the case of segments) or how they can be addressed (in the case of fragments). A manifest file,

or set of manifest files, may also identify the different encodings, and thus Chunk Collections, the content is available in.

In general, a HAS client will first request and receive a manifest file, and then, after parsing the information in the manifest file, proceed with sequentially requesting the chunks listed in the manifest file. Each HAS implementation has its own manifest file format and even within a particular format there are different methods available to specify the location of a chunk.

Of course managing the location of files is a core aspect of every CDN, and each CDN will have its own method of doing so. Some CDNs may be purely cache-based, with no higher-level knowledge of where each file resides at each instant in time. Other CDNs may have dedicated management nodes which, at each instant in time, do know at which servers each file resides. The CDNI interfaces designed in the CDNI WG will probably need to be agnostic to these kinds of CDN-internal architecture decisions. In the case of HAS there is a strict relationship between the location of the content in the CDN (in this case chunks) and the content itself (the locations specified in the manifest file). It is therefore useful to have an understanding of the different methods in use in CDNs today for specifying chunk locations in manifest files. The different methods for doing so are described in sections 2.2.1 to 2.2.3.

Although these sections are especially relevant for segmented content, due to its inherent distributed nature, the discussed methods are also applicable to fragmented content. Furthermore, it should be noted that the methods detailed below for specifying locations of content items in manifest files do not only relate to temporally segmented content (e.g. segments and fragments), but are also relevant in situations where content is made available in multiple qualities, encodings, or variants. In this case the content consists of multiple chunk collections, which may be described by either a single manifest file or multiple interrelated manifest files. In the latter case, there may be a high-level manifest file describing the various available bitrates, with URLs pointing to separate manifest files describing the details of each specific bitrate. For specifying the locations of the other manifest files, the same methods apply that are used for specifying chunk locations.

2.2.1. Full Locator

One method for specifying locations of chunks (or other manifest files) in a manifest file is through the use of a Full Locator. A Full Locator takes the form of an URL and is defined by the fact that it directly points to the specific chunk on the actual the server that is expected to deliver the requested chunk to the client.

An example of a Full Locator is the following:

```
http://deliverynode.server.cdn.com/content_1/segments/  
segment1_1.ts
```

As can be seen from this example URL, the URL includes both the identifier of the requested segment (in this case segment1_1.ts), as well as the server that is expected to deliver the segment (in this case deliverynode.server.cdn.com). With this, the client has enough information to directly request the specific segment from the specified delivery node.

2.2.2. Relative Locator

Another method for specifying chunk locations in a manifest file is through the use of Relative Locator. A Relative Locator is a pointer that is relative to the location where the manifest file has been acquired from. In most cases a Relative Locator will take the form of a string that has to be appended to the location of the manifest file to get the location of a specific chunk. This means that in the case a manifest with a Relative Locator is used, all chunks will be delivered by the same delivery node that delivered the manifest file. A Relative Locator will therefore not include a hostname.

For example, in the case a manifest file has been requested (and received) from `http://deliverynode.server.cdn.com/content_1/manifest.xml`, a Relative Locator pointing to a specific segment referenced in the manifest might be:

```
segments/segment1_1.ts
```

Which means that the client should take the location of the manifest file and append the Relative Locator. In this case, the segment would then be requested from `http://deliverynode.server.cdn.com/content_1/segments/segment1_1.ts`

2.2.3. Chunk Request Routing

A final method for specifying chunk locations in a manifest file is through the use of request routing. In this case, chunks are handled by the request routing system of a CDN just as if they were 'normal' content items. A chunk request comes in at a central request routing node and is handled just as if it were a regular content request, which might include looking up the delivery node best suited for delivering the requested chunk to the particular user and sending an HTTP redirect to the user with the URL pointing to the requested chunk on the specified delivery node.

An example of an URL pointing to a redirection mechanism might look as follows:

```
http://requestrouting.cdn.com/  
content_request?content=content_1&segment=segment1_1.ts
```

As can be seen from this example URL, the URL includes a pointer to a general CDN request routing function and includes some arguments identifying the requested segment.

3. Impact of HAS on CDNI

In the previous chapter, some of the unique properties of HAS have been discussed. Furthermore, some of the CDN-specific design decision with regards to addressing chunks have been detailed. In this chapter, the impact of supporting HAS in CDNI will be discussed. The scope of this chapter is explicitly not to define solutions, but merely to identify potential problems and issues that need to be agreed on. For this purpose, each subsection will pose a number of open questions that will need to be answered by the CDNI WG. At a later stage, the answers to these questions may be used to solicit HAS-related requirements for the CDNI Interfaces.

The chapter is divided into three subsections. The first subsection, 3.1, will discuss the impact supporting HAS has on the general CDNI architecture, use cases and requirements. The other four subsections, 3.2 to 3.5, will discuss the impact of HAS on each of the four CDNI Interfaces.

3.1. General

This section will discuss the impact supporting HTTP Adaptive Streaming has on the general CDNI architecture, use cases and requirements.

3.1.1. On the definition of a Content Item in CDNI

[I-D.ietf-cdni-problem-statement] defines content as

Content: Any form of digital data. One important form of content with additional constraints on distribution and delivery is continuous media (i.e. where there is a timing relationship between source and sink).

This very broad definition of content is useful for generalizing the CDNI interfaces in a way that allows them to be agnostic to the type of content that is delivered. However, what a Content Item in a CDN

constitutes may become relevant in the context of HAS if one considers a Content Item to be the element with which Content Metadata is associated, and the element that is the object of a CDN(I) request routing operation.

An example of a Content Item in the general sense is a video file or stream, such as a TV show or movie, or an audio file, such as an MP3. In a simple case, a single MP3 is a single Content Item. In a more complex case, a particular piece of content is made available in multiple resolutions, languages or qualities. A video item, for example, might be made available in three different resolutions. In these cases, it depends on the datamodel of a particular CDN (or a particular Content Provider) how it defines a Content Item. In some CDNs, all three video files might be seen as separate Content Items, each with their own set of Content Metadata. In other CDNs, the three alternative encodings of the same content are seen as a single Content Item, with a single set of Content Metadata describing that the content consists of three different versions. The CDNI Interfaces defined in the CDNI WG are affected by these kinds of differences in the ways different CDNs work and might need to be agnostic to these kinds of CDN-internal (or even Content Provider-related) decisions.

For content delivered using HAS, there is an even wider variety in the way different CDNs might interpret the definition of a Content Item. For example, CDNs using the Relative Locator method (see section 2.2.2) in their manifest files, might define all chunks that are part of the same Content Collection, and therefore referenced in the same (set of) manifest file(s), as a single Content Item for the purposes of Content Metadata and request routing. Other CDNs might define all chunks related to a single encoding of a particular video item, and thus part of the same Chunk Collection, as a single Content Item, thereby having multiple inter-related Content Items which are part of the same 'parent' Content Item. Yet another group of CDNs, especially those using the Chunk Request Routing method (see Section 2.2.3), might define every individual chunk as a separate Content Item, with a separate set of metadata describing each chunk.

In order for the CDNI WG to realise a standardized method of dealing with metadata, logging and request routing, it will be important to first have a common understanding of the term Content Item, and what it constitutes, especially with regard to HAS. One option would be to not impose a specific model, but allow the CDNI interfaces to support all the different definitions of Content Item (i.e. from considering each chunk to be a Content Item to considering all chunks originating from the same Original Content, and thus part of the same Content Collection, to be a single content item).

- o Should the CDNI Interfaces be agnostic to the definition of a Content Item in a particular CDN?

And if this is not the case, then

- o What constitutes a Content Item for the purposes of associating metadata and request routing?
- o How does the definition of a Content Item relate to Chunks, Chunk Collections and Content Collection?

If the WG decides to not impose a specific definition of what constitutes a Content Item, it is for further study whether it is required for all parties involved in the delivery of a single video item, CSP, uCDN and dCDN, to use the same definition. For example, is it possible for the uCDN to define a complete Content Collection as a single Content Item, but for the dCDN which delivers the actual content to see each chunk as a separate Content Item and handle the metadata accordingly?

- o Is it necessary for all CDNs involved in the delivery of a single video item to use the same definition of a Content Item (e.g. can the uCDN define a Content Collection as a Content Item and the dCDN define a chunk as a Content Item)?

3.1.2. General CDNI-HAS Requirements

This section is a placeholder for HAS-specific CDNI requirements that are not related to a specific CDNI interface.

3.2. Impact on Request Routing Interface

This section will discuss the impact supporting HTTP Adaptive Streaming has on the CDNI Request Routing Interface.

3.2.1. Dealing with manifest files

In section 2.2, three different methods for identifying and addressing chunks from within a manifest file were described: Full Locators, Relative Locators and Chunk Request Routing. Of course not every current CDN will use and/or support all three methods. Some CDNs may only support one of the three methods, while others may support two or all three.

The question is whether all CDNs involved in the delivery of a single video item need to support the same method. Is it for example possible for a dCDN to use Chunk Request Routing while the uCDN uses Full Locators? This question boils down to the more fundamental

question of who manages manifest file. Should a dCDN be allowed to change a manifest file? Or even create a new one?

- o Should the CDNI Interfaces be agnostic to the way chunks are identified in the manifest file and requested by the client?
- o Should it be possible for a uCDN and dCDN to use two different methods of addressing chunks in manifest files?

And, related to this

- o Should a dCDN be able to adapt a manifest file (i.e. is a manifest file part of the content, and therefore by definition not adaptable, or is it part of the delivery method) ?

If the CDNI WG decides that the answer to these questions is negative, this will probably mean that the only supported method for Chunk Addressing is using Relative Locators. For both Full Locators as well as Chunk Request Routing, it is necessary for the delivering CDN to change the URLs specified in the manifest file.

3.2.2. HAS Request Routing

One of the essential questions relating to HAS and request routing is whether CDNI request routing is handled on a per chunk level, a per Content Collection level, or somewhere in between. This question is tightly related to the definition of a Content Item, discussed in section 3.1.1. If a Content Item is the object of request routing, then it depends on the definition of a Content Item what type of content element is being request routed.

- o Should the CDNI Interfaces specify what element of a HAS stream is being request routed (i.e. should the CDNI interfaces support per-chunk request routing) ?

While having a separate request routing operation for every chunk will probably not be very efficient, only allowing for entire Content Collections to be request routed is very limiting. For example, it might be efficient for a dCDN targeted at mobile devices to only host (and thus be able to deliver) the lower resolution encodings of a given video item. In this case it probably wouldn't make sense to force the mobile CDN to host all resolutions (including the very high ones with multi-channel audio) that are hosted by the uCDN since there will be no clients accessing the high resolution content through the mobile CDN.

- o Should it be possible for a dCDN to host (and deliver) only a subset of all the chunks, or chunk collections, of a given Content Collection?

3.2.3. Dividing content over multiple nodes

An aspect that is related to the issues discussed in sections 3.2.1 and 3.2.2, is that of dividing content over multiple delivery nodes. In non-cache-based CDNs, where the location of content on delivery nodes is managed by a centralized process and where content is often pre-positioned, different Chunk Collections belonging to the same Content Collection, or even different chunks belonging to the same Chunk Collection, may be distributed over different delivery nodes. For example, the most popular resolutions of a particular Content Collection may be hosted on more delivery nodes than the less popular resolutions. In order to allow for this kind of internal CDN optimization it is necessary that the dCDN is able adapt the manifest file.

- o Should it be possible for a dCDN to distribute the chunks, or Chunk Collections, constituting a given Content Collection over its delivery nodes?

3.2.4. HAS Requirements for the CDNI Request Routing Interface

This section is a placeholder for HAS-specific requirements for the CDNI Request Routing interface.

3.3. Impact on Metadata Interface

This section will discuss the impact supporting HTTP Adaptive Streaming has on the CDNI Metadata Interface.

3.3.1. HAS-specific Metadata

In section 3.1.1, the impact of HAS on the definition of a Content Item, and what constitutes a Content Item was discussed. More specifically, it was discussed how different CDNs might see chunks or chunk collections as Content Items or as parts of Content Items. This question also has an effect on the way the CDNI Metadata Interface. If one defines a Content Item as the CDN element with which Content Metadata is associated, this raises the question how Content Metadata is associated with HAS content. For example, is there specific metadata element associated with each chunk, with each chunk collection, with each content collection, or is there a specific form of metadata for HAS content?

- o Is Content Metadata associated with Content Collections, Chunk Collections or chunks?
- o Is it necessary to extend the CDNI Metadata model with HAS-specific extensions?

3.3.2. HAS Requirements for the CDNI Metadata Interface

This section is a placeholder for HAS-specific requirements for the CDNI Metadata interface.

3.4. Impact on Logging Interface

This section will discuss the impact supporting HTTP Adaptive Streaming has on the CDNI Logging Interface.

3.4.1. Log processing

One aspect of using HAS in a CDN context that has been getting a lot of attention is logging. In contrast to other streaming solutions which are either session-based (e.g. RTP) or using a single file (e.g. Progressive Download), the chunked nature of HAS means that regular logging methods that simply log each content request will generate extremely large log files with a separate entry for each chunk being accessed. Apart from the large file size of these log files, a further problem is that due to the distributed nature of these log entries it can be difficult to trace them back to a specific number of clients or users, which makes reporting difficult.

For this reason, some CDNs use dedicated log processing software which accumulates, processes and aggregates log files. This log processing software is a further element where different CDNs might have different approaches. For example, some CDNs might do the log processing at a low-level, such as real-time in the delivery nodes. Other CDNs might process the log files in batches at a centralized location, such as once every hour or every day.

For the CDNI Logging interface, it will be necessary to realise a common understanding on what type of logging information is passed between the uCDN and the dCDN in the case a HAS-like protocol is used for delivery. Ideally, this interface is agnostic to the CDN-internal method of log processing. However, this might not be possible. For example, it can be argued that for transparency reasons, it is necessary for the dCDN to communicate the raw log files back to the uCDN. On the other hand, this creates a very large overhead in the inter-CDN communication, with log files for popular content possibly exceeding the file size of the content itself. Another method would be to require the dCDN to aggregate the log

files before reporting back to the uCDN, however this would require the dCDN to have specific log processing software.

- o Should the CDNI Logging Interface define a specific log-file format to be used for HAS content?

3.4.2. HAS Requirements for the CDNI Logging Interface

This section is a placeholder for HAS-specific requirements for the CDNI Logging interface.

3.5. Impact on Control Interface

This section will discuss the impact supporting HTTP Adaptive Streaming has on the CDNI Control Interface.

NOTE: At this point the impact of HAS on the CDNI Control Interface has not yet been determined.

3.5.1. HAS Requirements for the CDNI Control Interface

This section is a placeholder for HAS-specific requirements for the CDNI Control interface.

4. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

5. Security Considerations

TBD.

6. References

6.1. Normative References

[I-D.ietf-cdni-problem-statement]
Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement, draft-ietf-cdni-problem-statement-03", January 2012.

[I-D.ietf-cdni-use-cases]

Bertrand, G., Ed., Stephan, E., Watson, G., Burbridge, T.,
Eardley, P., and K. Ma, "Use Cases for Content Delivery
Network Interconnection, draft-ietf-cdni-use-cases-03",
January 2012.

6.2. Informative References

[Anchor] "".

Authors' Addresses

Ray van Brandenburg
TNO
Brassersplein 2
Delft 2612CT
the Netherlands

Phone: +31-88-866-7000
Email: ray.vanbrandenburg@tno.nl

M. Oskar van Deventer
TNO
Brassersplein 2
Delft 2612CT
the Netherlands

Phone: +31-88-866-7000
Email: oskar.vandeventer@tno.nl

Network Working Group
Internet-Draft
Intended status: Informational
Expires: October 14, 2013

R. van Brandenburg
O. van Deventer
TNO
F. Le Faucheur
K. Leung
Cisco Systems
April 12, 2013

Models for adaptive-streaming-aware CDN Interconnection
draft-brandenburg-cdni-has-05

Abstract

This document presents thoughts on the potential impact of supporting HTTP Adaptive Streaming technologies in CDN Interconnection (CDNI) scenarios. The intent is to present the authors' analysis of the CDNI-HAS problem space and discuss different options put forward both by the authors (and by others during informal discussions) on how to deal with HAS in the context of CDNI. This document has been used as input information during the WG process for making its decision regarding support for HAS.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 14, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Terminology	4
2. HTTP Adaptive Streaming aspects relevant to CDNI	5
2.1. Segmentation versus Fragmentation	5
2.2. Addressing chunks	6
2.2.1. Relative URLs	7
2.2.2. Absolute URLs with Redirection	8
2.2.3. Absolute URL without Redirection	9
2.3. Live vs. VoD	10
2.4. Stream splicing	11
3. Possible HAS Optimizations	11
3.1. File Management and Content Collections	12
3.1.1. General Remarks	12
3.1.2. Candidate approaches	12
3.1.2.1. Option 1.1: No HAS awareness	12
3.1.2.2. Option 1.2: Allow single file storage of fragmented content	13
3.1.2.3. Option 1.3: Access correlation hint	14
3.1.3. Recommendation	14
3.2. Content Acquisition of Content Collections	14
3.2.1. General Remarks	14
3.2.2. Candidate Approaches	15
3.2.2.1. Option 2.1: No HAS awareness	15
3.2.2.2. Option 2.2: Allow single file acquisition of fragmented content	16
3.2.3. Recommendation	16
3.3. Request Routing of HAS content	17
3.3.1. General remarks	17
3.3.2. Candidate approaches	17
3.3.2.1. Option 3.1: No HAS awareness	17
3.3.2.2. Option 3.2: Manifest File rewriting by uCDN	19
3.3.2.3. Option 3.3: Two-step Manifest File rewriting	20
3.3.3. Recommendation	22
3.4. Logging	22
3.4.1. General remarks	22
3.4.2. Candidate Approaches	23
3.4.2.1. Option 4.1: "Do-Nothing" Approach	23
3.4.2.2. Option 4.2: "CDNI Metadata Content Collection ID" Approach	25

3.4.2.3.	Option 4.3: "CDNI Logging Interface Compression"	
	Approach	26
3.4.2.4.	Option 4.4: "Full HAS awareness/per-Session-Logs"	
	Approach	27
3.4.3.	Recommendation	28
3.5.	URL Signing	30
3.5.1.	HAS Implications	30
3.5.2.	CDNI Considerations	31
3.5.3.	Option 5.1: Do Nothing	32
3.5.4.	Option 5.2: Flexible URL Signing by CSP	32
3.5.5.	Option 5.3: Flexible URL Signing by Upstream CDN	35
3.5.6.	Option 5.4: Authorization Group ID and HTTP Cookie	35
3.5.7.	Option 5.5: HAS-awareness with HTTP Cookie in CDN	36
3.5.8.	Option 5.6: HAS-awareness with Manifest in CDN	38
3.5.9.	Recommendation	38
3.6.	Content Purge	39
3.6.1.	Option 6.1: No HAS awareness	40
3.6.2.	Option 6.2: Purge Identifiers	40
3.6.3.	Recommendation	41
3.7.	Other issues	41
4.	IANA Considerations	41
5.	Security Considerations	41
6.	Acknowledgements	42
7.	References	42
7.1.	Normative References	42
7.2.	Informative References	42
	Authors' Addresses	42

1. Introduction

[RFC6707] defines the problem space for CDN Interconnection (CDNI) and the associated CDNI interfaces. This includes support, through interconnected CDNs, of content delivery to endusers using HTTP progressive download and HTTP Adaptive Streaming (HAS).

HTTP Adaptive Streaming is an umbrella term for various HTTP-based streaming technologies that allow a client to adaptively switch between multiple bitrates depending on current network conditions. A defining aspect of HAS is that, since it is based on HTTP, it is a pull-based mechanism, with a client actively requesting content segments, instead of the content being pushed to the client by a server. Due to this pull-based nature, media servers delivering content using HAS often show different characteristics when compared with media servers delivering content using traditional streaming methods such as RTP/RTSP, RTMP and MMS.

This document presents a discussion on what the impact of these different characteristics is to the CDNI interfaces and what HAS-specific optimizations may be required or may be desirable. The scope of this document is to present the authors' analysis of the CDNI-HAS problem space and discuss different options put forward both by the authors (and by others during informal discussions) on how to deal with HAS in the context of CDNI. The document concludes by presenting the authors' recommendations on how the CDNI WG should deal with HAS in its initial charter, with a focus on 'making it work' instead of including 'nice-to-have' optimizations that might delay the development of the CDNI WG deliverables identified in its initial charter.

It should be noted that the document is not a WG document, but has been used as input information during the WG process for making its decision regarding support for HAS. We expect the analysis presented in the document will also be useful in the future if and when the WG re-charters and wants to re-assess the level of HAS optimizations to be supported in CDNI scenarios.

1.1. Terminology

This document uses the terminology defined in [RFC6707] and [I-D.ietf-cdni-framework].

For convenience, the definition of HAS-related terms are restated here:

Content Item: A uniquely addressable content element in a CDN. A content item is defined by the fact that it has its own Content Metadata associated with it. It is the object of a request routing operation in a CDN. An example of a Content Item is a video file/stream, an audio file/stream or an image file.

Chunk: a fixed length element that is the result of a segmentation or fragmentation operation and that is independently addressable.

Fragment: A specific form of chunk (see Section 2.1). A fragment is stored as part of a larger file that includes all chunks that are part of the Chunk Collection.

Segment: A specific form of chunk (see Section 2.1). A segment is stored as a single file from a file system perspective.

Original Content: Non-chunked content that is the basis for a segmentation or fragmentation operation. Based on Original Content, multiple alternative representations (using different encoding methods, supporting different resolutions and/or targeting different

bitrates) may be derived, each of which may be fragmented or segmented.

Chunk Collection: The set of all chunks that are the result of a single segmentation or fragmentation operation being performed on a single representation of the Original Content. A Chunk Collection is described in a Manifest File.

Content Collection: The set of all Chunk Collections that are derived from the same Original Content. A Content Collection may consist of multiple Chunk Collections, each corresponding to a single representation of the Original Content. A Content Collection may be described by one or more Manifest Files.

Manifest File: A Manifest File, also referred to as Media Presentation Description (MPD) file, is a file that list the way the content has been chunked (possibly for multiple encodings) and where the various chunks are located (in the case of segments) or how they can be addressed (in the case of fragments).

2. HTTP Adaptive Streaming aspects relevant to CDNI

In the last couple of years, a wide variety of HAS-like protocols have emerged. Among them are proprietary solutions such as Apple's HTTP Live Streaming (HLS), Microsoft's HTTP Smooth Streaming (HSS) and Adobe's HTTP Dynamic Streaming (HDS), and various standardized solutions such as 3GPP Adaptive HTTP Streaming (AHS) and MPEG Dynamic Adaptive Streaming over HTTP (DASH). While all of these technologies share a common set of features, each has its own defining elements. This chapter will look at some of the common characteristics and some of the differences between these technologies and how those might be relevant to CDNI. In particular, Section 2.1 will describe the various methods to store HAS content and Section 2.2 will list three methods that are used to address HAS content in a CDN. After these generic HAS aspects are discussed, two special situations that need to be taken into account when discussing HAS are addressed: Section 2.3 discusses the differences between Live and VoD content, while Section 2.4 discusses the scenario where multiple streams are combined in a single Manifest File (e.g. for ad insertion purposes).

2.1. Segmentation versus Fragmentation

All HAS implementations are based around a concept referred to as chunking: the concept of having a server split content up in numerous fixed duration chunks, which are independently decodable. By sequentially requesting and receiving chunks, a client can recreate and play out the content. An advantage of this mechanism is that it allows a client to seamlessly switch between different encodings of

the same Original Content at chunk boundaries. Before requesting a particular chunk, a client can choose between multiple alternative encodings of the same chunk, irrespective of the encoding of the chunks it has requested earlier.

While every HAS implementation uses some form of chunking, not all implementations store the resulting chunks in the same way. In general, there are two distinct methods of performing chunking and storing the results: segmentation and fragmentation.

- With segmentation, which is for example mandatory in all versions of Apple's HLS prior to version 7, the chunks, in this case also referred to as segments, are stored completely independent from each other, with each segment being stored as a separate file from a file system perspective. This means that each segment has its own unique URL with which it can be retrieved.
- With fragmentation (or virtual segmentation), which is for example used in Microsoft's Smooth Streaming, all chunks, or fragments, belonging to the same Chunk Collection are stored together, as part of a single file. While there are a number of container formats which allow for storing this type of chunked content, Fragmented MP4 is most commonly used. With fragmentation, a specific chunk is addressable by subfixing the common file URL with an identifier uniquely identifying the chunk that one is interested in, either by timestamp, by byterange, or in some other way.

While one can argue about the merits of each of these two different methods of handling chunks, both have their advantages and drawbacks in a CDN environment. For example, fragmentation is often regarded as a method that introduces less overhead, both from a storage and processing perspective. Segmentation on the other hand, is regarded as being more flexible and easier to cache. In practice, current HAS implementations increasingly support both methods.

2.2. Addressing chunks

In order for a client to request chunks, either in the form of segments or in the form of fragments, it needs to know how the content has been chunked and where to find the chunks. For this purpose, most HAS protocols use a concept that is often referred to as a Manifest File (also known as Media Presentation Description, or MPD); i.e. a file that lists the way the content has been chunked and where the various chunks are located (in the case of segments) or how they can be addressed (in the case of fragments). A Manifest File, or set of Manifest Files, may also identify the different representations, and thus Chunk Collections, available for a content.

In general, a HAS client will first request and receive a Manifest File, and then, after parsing the information in the Manifest File, proceed with sequentially requesting the chunks listed in the Manifest File. Each HAS implementation has its own Manifest File format and even within a particular format there are different methods available to specify the location of a chunk.

Of course managing the location of files is a core aspect of every CDN, and each CDN will have its own method of doing so. Some CDNs may be purely cache-based, with no higher-level knowledge of where each file resides at each instant in time. Other CDNs may have dedicated management nodes which, at each instant in time, do know at which servers each file resides. The CDNI interfaces designed in the CDNI WG will probably need to be agnostic to these kinds of CDN-internal architecture decisions. In the case of HAS there is a strict relationship between the location of the content in the CDN (in this case chunks) and the content itself (the locations specified in the Manifest File). It is therefore useful to have an understanding of the different methods in use in CDNs today for specifying chunk locations in Manifest Files. The different methods for doing so are described in sections 2.2.1 to 2.2.3.

Although these sections are especially relevant for segmented content, due to its inherent distributed nature, the discussed methods are also applicable to fragmented content. Furthermore, it should be noted that the methods detailed below for specifying locations of content items in Manifest Files do not only relate to temporally segmented content (e.g. segments and fragments), but are also relevant in situations where content is made available in multiple representations (e.g., in different qualities, encoding methods, resolutions and/or bitrates). In this case the content consists of multiple chunk collections, which may be described by either a single Manifest File or multiple interrelated Manifest Files. In the latter case, there may be a high-level Manifest File describing the various available bitrates, with URLs pointing to separate Manifest Files describing the details of each specific bitrate. For specifying the locations of the other Manifest Files, the same methods apply that are used for specifying chunk locations.

One final note relates to the delivery of the Manifest Files themselves. While in most situations the delivery of both the Manifest File and the chunks are handled by the CDN, there are scenarios imaginable in which the Manifest File is delivered by e.g. the Content Provider, and the Manifest File is therefore not visible to the CDN.

2.2.1. Relative URLs

One method for specifying chunk locations in a Manifest File is through the use of relative URLs. A relative URL is a URL that does not include the HOST part of a URL but only includes (part of) the PATH part of a URL. In practice, a relative URL is used by the client as being relative to the location where the Manifest File has been acquired from. In these cases a relative URL will take the form of a string that has to be appended to the location of the Manifest File to get the location of a specific chunk. This means that in the case a Manifest File with relative URLs is used, all chunks will be delivered by the same surrogate that delivered the Manifest File. A relative URL will therefore not include a hostname.

For example, in the case a Manifest File has been requested (and received) from:

```
http://surrogate.server.cdn.example.com/content_1/manifest.xml
```

, a relative URL pointing to a specific segment referenced in the Manifest File might be:

```
segments/segment1_1.ts
```

Which means that the client should take the location of the Manifest File and append the relative URL. In this case, the segment would then be requested from `http://surrogate.server.cdn.example.com/content_1/segments/segment1_1.ts`

The downside of using relative URLs is that it forces a CDN to deliver all segments belonging to a given content item with the same surrogate that delivered the Manifest File for that content item, which results in limited flexibility. Another drawback is that Relative URLs do not allow for fallback URLs; should the surrogate that delivered the Manifest File break down, the client is no longer able to request chunks. The advantage of relative URLs is that it is very easy to transfer content between different surrogates and even CDNs.

2.2.2. Absolute URLs with Redirection

Another method for specifying locations of chunks (or other Manifest Files) in a Manifest File is through the use of an absolute URL. An absolute URL contains a fully formed URL (i.e. the client does not have to calculate the URL as in the case of the relative URL but can use the URL from the Manifest File directly).

In the context of Manifest Files, there are two types of absolute URLs imaginable: Absolute URLs with Redirection and Absolute URLs without Redirection. The two methods differ in whether the URL

points to a request routing node which will redirect the client to a surrogate (Absolute URL with Redirection) or point directly to a surrogate hosting the requested content (Absolute URL without Redirection).

In the case of Absolute URLs with Redirection, a request for a chunk is handled by the request routing system of a CDN just as if it were a standalone (non-HAS) content request, which might include looking up the surrogate (and/or CDN) best suited for delivering the requested chunk to the particular user and sending an HTTP redirect to the user with the URL pointing to the requested chunk on the specified surrogate (and/or CDN), or a DNS response pointing to the specific surrogate.

An example of an Absolute URL with Redirection might look as follows:

```
http://requestrouting.cdn.example.com/  
content_request?content=content_1&segment=segment1_1.ts
```

As can be seen from this example URL, the URL includes a pointer to a general CDN request routing function and includes some arguments identifying the requested segment.

The advantage of using Absolute URLs with Redirection is that it allows for maximum flexibility (since chunks can be distributed across surrogates and CDN in any imaginable way) without having to modify the Manifest File every time one or more chunks are moved (as is the case when Absolute URLs without Redirection are used). The downside of this method is that it can add significant load to a CDN request routing system, since it has to perform a redirect every time a client requests a new chunk.

2.2.3. Absolute URL without Redirection

In the case of the Absolute URL without Redirection, the URL points directly to the specific chunk on the actual surrogate that will deliver the requested chunk to the client. In other words, there will be no HTTP redirection operation taking place between the client requesting the chunk and the chunk being delivered to the client by the surrogate.

An example of an Absolute URL without Redirection is the following:

```
http://surrogate.cdn.example.com/content_1/segments/segment1_1.ts
```

As can be seen from this example URL, the URL includes both the identifier of the requested segment (in this case segment1_1.ts), as well as the server that is expected to deliver the segment (in this

case surrogate.cdn.example.com). With this, the client has enough information to directly request the specific segment from the specified surrogate.

The advantage of using Absolute URLs without Redirection is that it allows more flexibility compared to using Relative URLs (since segments do not necessarily have to be delivered by the same server) while not requiring per-segment redirection (which would add significant load to the node doing the redirection). The drawback of Absolute URLs without Redirection is that it requires a modification of the Manifest File every time content is moved to a different location (either within a CDN or across CDNs).

2.3. Live vs. VoD

Though the formats and addresses of Manifest Files and chunk files do not typically differ significantly between live and Video-on-Demand (VoD) content, the time at which the Manifest Files and chunk files become available does differ significantly. For live content, chunk files and their corresponding Manifest Files are created and delivered in real-time. This poses a number of potential issues for HAS optimization:

- With live content, chunk files are made available in real-time. This limits the applicability of bundling for content acquisition purposes. Prepositioning may still be employed, however, any significant latency in the prepositioning may diminish the value of prepositioning if a client requests the chunk prior to prepositioning, or if the prepositioning request is serviced after the chunk playout time has passed.
- In the case of live content, Manifest Files must be updated for each chunk and therefore must be retrieved by the client prior to each chunk request. Any Manifest-File based optimization schemes must therefore be prepared to optimize on a per-segment request basis. Manifest Files may also be polled multiple times prior to the actual availability of the next chunk.
- Since live Manifest Files are updated as each new chunk becomes available, the cacheability of Manifest Files is limited. Though timestamping and reasonable TTLs can improve delivery performance, timely replication and delivery of updated Manifest Files is critical to ensuring uninterrupted playback.

- Manifest Files are typically updated after the corresponding chunk is available for delivery, to prevent premature requests for chunks which are not yet available. HAS optimization approaches which employ dynamic Manifest File generation must be synchronized with chunk creation to prevent playback errors.

2.4. Stream splicing

Stream splicing is used to create media mashups, combining content from multiple sources. A common example in which content resides outside the CDNs is with advertisement insertion, for both VoD and live streams. Manifest Files which contain Absolute URLs with redirection may contain chunk or nested Manifest File URLs which point to content not delivered via any of the interconnected CDNs.

Furthermore, client and downstream proxy devices may depend on non-URL information provided in the Manifest File (e.g., comments or custom tags) for performing stream splicing. This often occurs outside the scope of the interconnected CDNs. HAS optimization schemes which employ dynamic Manifest File generation or rewriting must be cognizant of chunk URLs, nested Manifest File URLs, and other metadata which should not be modified or removed. Improper modification of these URLs or other metadata may cause playback interruptions, and in the case of unplayed advertisements, may result in loss of revenue for content providers.

3. Possible HAS Optimizations

In the previous chapter, some of the unique properties of HAS have been discussed. Furthermore, some of the CDN-specific design decisions with regards to addressing chunks have been detailed. In this chapter, the impact of supporting HAS in CDN Interconnection scenarios will be discussed.

There are a number of topics, or problem areas, that are of particular interest when considering the combination of HAS and CDNI. For each of these problem areas it holds that there are a number of different ways in which the CDNI Interfaces can deal with them. In general it can be said that each problem area can either be solved in a way that minimizes the amount of HAS-specific changes to the CDNI Interfaces or in way that maximizes the flexibility and efficiency with which the CDNI Interfaces can deliver HAS content. The goal for the CDNI WG should probably be to try to find the middle ground between these two extremes and try to come up with solutions that optimize the balance between efficiency and additional complexity.

In order to allow the WG to make this decision, this chapter will briefly describe each of the following problem areas together with a

number of different options for dealing with them. Section 3.1 will discuss the problem of how to deal with file management of groups of files, or Content Collections. Section 3.2 will deal with a related topic: how to do content acquisition of Content Collections between the uCDN and dCDN. After that, Section 3.3 describes the various options for the request routing of HAS content, particularly related to Manifest Files. Section 3.4 talks about a number of possible optimizations for the logging of HAS content, while Section 3.5 discusses the options regarding URL signing. Section 3.6 finally, describes different scenarios for dealing with the removal of HAS content from CDNs.

3.1. File Management and Content Collections

3.1.1. General Remarks

One of the unique properties of HAS content is that it does not consist of a single file or stream but of multiple interrelated files (segment, fragments and/or Manifest Files). In this document this group of files is also referred to as a Content Collection. Another important aspect is the difference between segments and fragments (see Section 2.1).

Irrespective of whether segments or fragments are used, different CDNs might handle Content Collections differently from a file management perspective. For example, some CDNs might handle all files belonging to a Content Collection as individual files, which are stored independently from each other. An advantage of this approach is that makes it easy to cache individual chunks. Other CDNs might store all fragments belonging to a Content Collection in a bundle, as if they were a single file (e.g. by using a fragmented MP4 container). The advantage of this approach is that it reduces file management overhead.

This section will look at the various ways with which the CDNI interfaces might deal with these differences in handling Content Collections from a file management perspective. The different options can be distinguished based on the level of HAS-awareness they require on the part of the different CDNs and the CDNI interfaces.

3.1.2. Candidate approaches

3.1.2.1. Option 1.1: No HAS awareness

This first option assumes no HAS awareness in both the involved CDNs and the CDNI Interfaces. This means that the uCDN uses individual files and the dCDN is not explicitly made aware of the relationship between chunks and it doesn't know which files are part of the same

Content Collection. In practice this scenario would mean that the file management method used by the uCDN is simply imposed on the dCDN as well.

This scenario also means that it is not possible for the dCDN to use any form of file bundling, such as the single-file mechanism which can be to store fragmented content as a single file (see Section 2.1). The one exception to this rule is the situation where the content is fragmented and the Manifest Files on the uCDN contains byte range requests, in which case the dCDN might be able to acquire fragmented content as a single file (see Section 3.2.2.2).

Effect on CDNI interfaces:

- o None

Advantages/Drawbacks:

- + No HAS awareness necessary in CDNs, no changes to CDNI Interfaces necessary
- The dCDN is forced to store chunks as individual files.

3.1.1.2.2. Option 1.2: Allow single file storage of fragmented content

In some cases, the dCDN might prefer to store fragmented content as a single file on its surrogates to reduce file management overhead. In order to do so, it needs to be able to either acquire the content as a single file (see Section 3.2.2.2), or merge the different chunks together and place them in the same container (e.g. fragmented MP4). The downside of this is that in order to do so, the dCDN needs to be fully HAS aware.

Effect on CDNI interfaces:

- o CDNI Metadata Interface: Add fields for indicating the particular type of HAS (e.g. MPEG DASH or HLS) that is used and whether segments or fragments are used
- o CDNI Metadata Interface: Add field for indicating the name and type of the Manifest File(s)

Advantages/Drawbacks:

- + Allows dCDN to store fragmented content as a single file, reducing file management overhead
- Complex operation, requiring dCDN to be fully HAS aware

3.1.2.3. Option 1.3: Access correlation hint

An intermediary approach between the two extremes detailed in the previous two sections is one that uses a 'Access Correlation Hint'. This hint, which is added to the CDNI Metadata of all chunks of a particular Content Collection, indicates that those files are likely to be requested in a short time window from each other. This information can help a dCDN to implement local file storage optimizations for VoD items (e.g. by bundling all files with the same Access Correlation Hint value in a single bundle/file), thereby reducing the number of files it has to manage while not requiring any HAS awareness.

Effect on CDNI interfaces:

- o CDNI Metadata Interface: Add field for indicating Access Correlation Hint

Advantages/Drawbacks:

- + Allows dCDN to perform file management optimization
- + Does not require any HAS awareness
- + Very small impact on CDNI Interfaces
- Expected benefit compared with Option 1.1 is small

3.1.3. Recommendation

Based on the listed pros and cons, the authors recommend the WG to go for Option 1.1, the 'Do Nothing'-approach. The likely benefits from going for Option 1.3 are not believed to be significant enough to warrant changing the CDNI Metadata Interface. Although Option 1.2 would bring definite benefits for HAS aware dCDNs, going for this options would require significant CDNI extensions that would impact the WG's milestones. The authors therefore don't recommend to include it in the current work but mark it as a possible candidate for rechartering once the initial CDNI solution is completed.

3.2. Content Acquisition of Content Collections

3.2.1. General Remarks

In the previous section the relationship between file management and HAS in a CDNI scenario has been discussed. This section will discuss a related topic, which is content acquisition between two CDNs.

With regards to content acquisition, it is important to note the difference between CDNs that do Dynamic Acquisition of content and CDNs that perform Content Pre-positioning. In the case of dynamic acquisition, a CDN only requests a particular content item when a cache-miss occurs. In the case of pre-positioning, a CDN proactively places content items on the nodes on which it expects traffic for that particular content item. For each of these types of CDNs, there might be a benefit in being HAS aware. For example, in the case of dynamic acquisition, being HAS aware means that after a cache miss for a given chunk occurs, that node might not only acquire the requested chunk, but might also acquire some related chunks that are expected to be requested in the near future. In the case of pre-positioning, similar benefits can be had.

3.2.2. Candidate Approaches

3.2.2.1. Option 2.1: No HAS awareness

This first option assumes no HAS awareness in both the involved CDNs and the CDNI Interfaces. Just as with Option 1.1 discussed in the previous section with regards to file management, having no HAS awareness means that the dCDN is not aware of the relationship between chunks. In the case of content acquisition, this means that each and every file belonging to a Content Collection will have to be individually acquired from the uCDN by the dCDN. The exception to the rule is in cases with fragmented content where the uCDN uses Manifest Files which contain byte range requests. In this case the dCDN can simply omit the byte range identifier and acquire the complete file.

The advantage of this approach is that it is highly flexible. If a client only requests a small portion of the chunks belonging to a particular Content Collection, the dCDN only has to acquire those chunks from the uCDN, saving both bandwidth and storage capacity.

The downside of acquiring content on a per-chunk basis is that it creates more transaction overhead between the dCDN and uCDN compared to a method in which entire Content Collections can be acquired as part of one transaction.

Effect on CDNI interfaces:

- o None

Advantages/Drawbacks:

- + Per-chunk content acquisition allows for high level of flexibility between dCDN and uCDN

- Per-chunk content acquisition creates more transaction overhead between dCDN and uCDN

3.2.2.2. Option 2.2: Allow single file acquisition of fragmented content

As discussed in Section 3.2.2.1, there is one (fairly rare) case where fragmented content can be acquired as a single file without any HAS awareness and that is when fragmented content is used and where the Manifest File specifies byte range request. This section discusses how to perform single file acquisition in the other (very common) cases. To do so, the dCDN would have to have full-HAS awareness (at least to the extent of being able to map between single file and individual chunks to serve).

Effect on CDNI interfaces:

- o CDNI Metadata Interface: Add fields for indicating the particular type of HAS (e.g. MPEG DASH or HLS) that is used and whether segments or fragments are used
- o CDNI Metadata Interface: Add field for indicating the name and type of the Manifest File(s)

Advantages/Drawbacks:

- + Allows for more efficient content acquisition in all HAS-specific supported forms
- Requires full HAS awareness on part of dCDN
- Requires significant CDNI Metadata Interface extensions

3.2.3. Recommendation

Based on the listed pros and cons, the authors recommend the WG to go for Option 2.1 since it is sufficient to 'make HAS work'. While Option 2.2 would bring benefits to the acquisition of large Content Collections, it would require significant CDNI extensions which would impact the WG's milestones. Option 2.2 might be a candidate to include in possible rechartering once the initial CDNI solution is completed.

3.3. Request Routing of HAS content

3.3.1. General remarks

In this section the effect HAS content has on request routing will be identified. Of particular interest in this case are the different types of Manifest Files that might be used. In Section 2.2, three different methods for identifying and addressing chunks from within a Manifest File were described: Relative URLs, Absolute URLs without Redirection and Absolute URLs with Redirection. Of course not every current CDN will use and/or support all three methods. Some CDNs may only use one of the three methods, while others may support two or all three.

An important factor in deciding which chunk addressing method is used is the Content Provider. Some Content Providers may have a strong preference for a particular method and deliver the Manifest Files to the CDN in a particular way. Depending on the CDN and the agreement it has with the Content Provider, a CDN may either host the Manifest Files as they were created by the Content Provider, or modify the Manifest File to adapt it to its particular architecture (e.g. by changing relative URLs to Absolute URLs which point to the CDN Request Routing function).

3.3.2. Candidate approaches

3.3.2.1. Option 3.1: No HAS awareness

This first option assumes no HAS awareness in both the involved CDNs and the CDNI Interfaces. This scenario also assumes that neither the dCDN nor the uCDN have the ability to actively manipulate Manifest Files. As was also discussed with regards to file management and content acquisition, having no HAS awareness means that each file constituting a Content Collections is handled on an individual basis, with the dCDN unaware of any relationship between files.

The only chunk addressing method that works without question in this case is Absolute URLs with Redirection. In other words, the Content Provider that ingested the content into the uCDN created a Manifest File with each chunk location pointing to the Request Routing function of the uCDN. Alternatively, the Content Provider may have ingested the Manifest File containing relative URLs and the uCDN ingestion function has translated these to Absolute URLs pointing to the Request Routing function.

In this Absolute URL with Redirection case, the uCDN can simply have the Manifest File be delivered by the dCDN as if it were a regular file. Once the client parses the Manifest File, it will request any

subsequent chunks from the uCDN Request Routing function. That function can then decide to outsource the delivery of that chunk to the dCDN. Depending on whether HTTP-based (recursive or iterative) or DNS-based request routing is used, the uCDN Request Routing function will then either directly or indirectly redirect the client to the Request Routing function of the dCDN (assuming it does not have the necessary information to redirect the client directly to a surrogate in the dCDN).

The drawback of this method is that it creates a large amount of request routing overhead for both the uCDN and dCDN. For each chunk the full inter-CDN Request Routing process is invoked (which can result in two HTTP redirections in the case of iterative redirection, or result in one HTTP redirection plus one CDNI Request Routing/Redirection Interface request/response). Even in the case where DNS-based redirection is used, there might be significant overhead involved since both the dCDN and uCDN Request Routing function might have to perform database lookups and query each other. While with DNS this overhead might be reduced by using DNS' inherent caching mechanism, this will have significant impact on the accuracy of the redirect.

With no HAS awareness, Relative URLs might or might not work depending on the type of Relative URL that is used. When a uCDN delegates the delivery of a Manifest File containing Relative URLs to a dCDN, the client goes directly to the dCDN surrogate from which it has received the Manifest File for every subsequent chunk. As long as the Relative URL is not path-absolute (see [RFC3986]), this approach will work fine.

Since using Absolute URLs without Redirection inherently require a HAS aware CDN, they also cannot be used in this case. The reason for this is that with Absolute URLs without Redirection, the URLs in the Manifest File will point directly to a surrogate in the uCDN. Since this scenario assumes no HAS awareness on the part of the dCDN or uCDN, it is impossible for either of these CDNs to rewrite the Manifest File and thus allow the client to either go to a surrogate in the dCDN or to a request routing function.

Effect on CDNI interfaces:

- o None

Advantages/Drawbacks:

- + Supports Absolute URLs with Redirection
- + Supports Relative URLs

- + Does not require HAS awareness and/or changes to the CDNI Interfaces
- Not possible to use Absolute URLs without Redirection
- Creates significant signaling overhead in case Absolute URLs with Redirection are used (inter-CDN request redirection for each chunk)

3.3.2.2. Option 3.2: Manifest File rewriting by uCDN

While Option 3.1 does allow for Absolute URLs with Redirection to be used, it does so in a way that creates a high-level of request routing overhead for both the dCDN and the uCDN. This option presents a solution to significantly reduce this overhead.

In this scenario, the uCDN is able to rewrite the Manifest File (or generate a new one) to be able to remove itself from the request routing chain for chunks being referenced in the Manifest File. As described in Section 3.3.2.1, in the case of no HAS awareness the client will go to the uCDN request routing function for each chunk request. This request routing function can then redirect the client to the dCDN request routing function. By rewriting the Manifest File (or generating a new one), the uCDN is able to remove this first step, and have the Manifest File point directly to the dCDN request routing function.

A key advantage of this solution is that it does not directly have an impact on the CDNI Interfaces and is therefore transparent to these interfaces. It is a CDN-internal function that a uCDN can perform autonomously by using information configured for regular CDNI operation or that is received from the dCDN as part of the regular communication using the CDNI Request Routing/Redirection Interface.

More specifically, in order for the uCDN to rewrite the Manifest File, the minimum information needed is the location of the dCDN request routing function (or alternatively the location of the dCDN delivering surrogate). This information can be available from configuration or can be derived from the regular CDNI Request Routing/Redirection Interface. For example, the uCDN may ask the dCDN for the location of its request routing node (through the CDNI Request Routing/Redirection Interface) every time a request for a Manifest File is received and processed by the uCDN request routing function. The uCDN would then modify the Manifest File and deliver the Manifest File to the client. One advantage of this method is that it maximizes efficiency and flexibility by allowing the dCDN to optionally respond with the locations of its surrogates instead of the location of its request routing function (and effectively turning

the URLs into Absolute URLs without Redirection). There are many variations around this approach, such as where the modification of the Manifest File is only performed once (or once per period of time) by the uCDN request routing function, when the first client for that particular Content Collection (and redirected to that particular dCDN) sends a Manifest File request. The advantage of such a variation is that the uCDN only has to modify the Manifest File once (or once per time period). The drawback of this variation is that the dCDN is no longer in a position to influence the request routing decision across individual content requests.

It should be noted that there are a number of things to take into account when changing a Manifest File (see for example Section 2.3 and Section 2.4 on live HAS content and ad insertion). Furthermore, some Content Providers might have issues with a CDN changing Manifest Files. However, in this option the Manifest File manipulation is only being performed by the uCDN, which can be expected to be aware of these limitations if it wants to perform Manifest File manipulation since it is in its own best interest that its customer's content gets delivered in the proper way and since there is a direct commercial and technical relationship between the uCDN (the Authoritative CDN in this scenario) and its customer (the Content Provider). Should the Content Provider want to limit Manifest File manipulation, it can simply arrange this with the uCDN bilaterally.

Effect on CDNI interfaces:

- o None

Advantages/Drawbacks:

- + Possible to significantly decrease signaling overhead when using Absolute URLs.
- + (Optional) Possible to have uCDN rewrite the Manifest File with locations of surrogates in dCDN (turning Absolute URLs with Redirection in Absolute URLs without Redirection)
- + No changes to CDNI Interfaces
- + Does not require HAS awareness in dCDN
- Requires high level of HAS awareness in uCDN (for modifying Manifest Files)

3.3.2.3. Option 3.3: Two-step Manifest File rewriting

One of the possibilities with Option 3.2 is allowing the dCDN to provide the locations of a specific surrogate to the uCDN, so that the uCDN can fit the Manifest File with Absolute URLs without Redirection and the client can request chunks directly from a dCDN surrogate. However, some dCDNs might not be willing to provide this information to the uCDN. In that case they can only provide the uCDN with the location of their request routing function and thereby preventing use of Absolute URLs without Redirection.

One method for solving this limitation is allowing two-step Manifest File manipulation. In the first step the uCDN would perform its own modification, and place the locations of the dCDN request routing function in the Manifest File. Then, once a request for the Manifest File comes in at the dCDN request routing function, it would perform a second modification in which it replaces the URLs in the Manifest Files with the URLs of its surrogates. This way the dCDN can still profit from having limited request routing traffic, while not having to share sensitive surrogate information with the uCDN.

The downside of this approach is that it not only assumes HAS awareness in the dCDN but it also requires some HAS-specific additions to the CDNI Metadata Interface. In order for the dCDN to be able to change the Manifest File, it has to have some information about the structure of the content. Specifically, it needs to have information about which chunks make up the Content Collection.

Effect on CDNI interfaces (apart from those already listed under Option 3.2):

- o CDNI Metadata Interface: Add necessary fields for conveying HAS specific information (e.g. the files that make up the Content Collection) to the dCDN.
- o CDNI Metadata Interface: Allow dCDN to modify Manifest File

Advantages/Drawbacks (apart from those already listed under Option 3.2):

- + Allows dCDN to use Absolute URLs without Redirection without having to convey sensitive information to the uCDN
- Requires high level of HAS awareness in dCDN (for modifying Manifest Files)
- Requires adding HAS-specific and Manifest File manipulation specific information to the CDNI Metadata Interface

3.3.3. Recommendation

Based on the listed pros and cons, the authors recommend to go for Option 3.1, with Option 3.2 as an optional feature that may be supported as a CDN-internal behavior by an uCDN. While Option 3.1 allows for HAS content to be delivered using the CDNI interfaces, it does so with some limitations regarding supported Manifest Files and, in some cases, with large signaling overhead. Option 3.2 can solve most of these limitations and presents a significant reduction of the request routing overhead. Since Option 3.2 does not require any changes to the CDNI interfaces but only changes the way the uCDN uses the existing interfaces, supporting it is not expected to result in a significant delay of the WG's milestones. The authors recommend the WG to not include Option 3.3, since it raises some questions of potential brittleness and including it would result in a significant delay of the WG's milestones.

3.4. Logging

3.4.1. General remarks

As stated in [RFC6707], "the CDNI Logging interface enables details of logs or events to be exchanged between interconnected CDNs".

As discussed in [I-D.draft-bertrand-cdni-logging], the CDNI logging information can be used for multiple purposes including maintenance/debugging by uCDN, accounting (e.g. in view of billing or settlement), reporting and management of end-user experience (e.g. to the CSP), analytics (e.g. by the CSP) and control of content distribution policy enforcement (e.g. by the CSP).

The key consideration for HAS with respect to logging is the potential increase of the number of Log records by two to three orders of magnitude, as compared to regular HTTP delivery of a video, since, by default, log records would typically be generated on a per-chunk-delivery basis instead of per-content-item-delivery basis. This impacts the scale of every processing step in the Logging Process (see [I-D.draft-bertrand-cdni-logging]), including:

- a. Logging information generation and storing on CDN elements (Surrogate, Request Routers, ...)
- b. Logging information aggregation within a CDN
- c. Logging information manipulation (including information protection, filtering, update and rectification)

- d. (Where needed) Logging information CDNI reformatting (e.g. reformatting from CDN-specific format to the CDNI Logging Interface format for export by dCDN to uCDN)
- e. Logging exchange via CDNI Logging Interface
- f. (Where needed) Logging re-reformatting (e.g. reformatting from CDNI Logging Interface format into log-consuming specific application)
- g. Logging consumption/processing (e.g. feed logs into uCDN accounting application, feed logs into uCDN reporting system to provide per CSP views, feed logs into debugging tools)

Note that there may be multiple instances of step [f] and [g] running in parallel.

While the CDNI Logging Interface is only used to perform step [e], we note that its format directly affects step [d] and [f] and that its format also indirectly affects the other steps (for example if the CDNI Logging Interface requires per-chunk log records, step [a], [b] and [d] cannot operate on a per-HAS-session basis and they also need to operate on a per-chunk basis).

This section discusses the main candidate approaches identified for CDNI in terms of dealing with HAS with respect to Logging.

3.4.2. Candidate Approaches

3.4.2.1. Option 4.1: "Do-Nothing" Approach

In this approach nothing is done specifically for HAS so that each HAS-chunk delivery is considered, for CDNI Logging, as a standalone content delivery. In particular, a separate log record for each HAS-chunk delivery is included in the CDNI Logging Interface in step [e] (as defined in Section 3.4.1). This approach requires that step [a], [b], [c], [d] and [e] also be performed on a per-chunk basis. This approach allows [g] to be performed either on a per-chunk basis (assuming step [f] maintains per-chunk records) or on a more "summarized" manner such as per-HAS-Session basis (assuming step [f] summarizes per-chunk records into per-HAS-session records).

Effect on CDNI interfaces:

- o None

Advantages/Drawbacks:

- + No information loss (i.e. all details of each individual chunk delivery are preserved). While this full level of detail may not be needed for some Log consuming applications (e.g. billing), this full level of detail is likely valuable (possibly required) for some Log consuming applications (e.g. debugging)
- + Easier integration (at least in the short term) into existing Logging tools since those are all capable of handling per-chunk records
- + No extension needed on CDNI interfaces
- High volume of logging information to be handled (storing & processing) at every step of the Logging process from [a] to [g] (while summarization in step [f] is conceivable, it may be difficult to achieve in practice without any hints for correlation in the log records).

An interesting question is whether a dCDN could use the CDNI Logging interface specified for the "Do-Nothing" approach to report summarized "per-session" log information in the case where the dCDN performs such summarization. The high level idea would be that, when a dCDN performs HAS log summarization for its own purposes anyways, this dCDN could include, in the CDNI Logging interface, one (or a few) log entry for a HAS session (instead of one entry per HAS-chunk) that summarizes the deliveries of many/all HAS-chunk for a session. However, the authors feel that, when considering the details of this, this is not achievable without explicit agreement between the uCDN and dCDN about how to perform/interpret such summarization. For example, when a HAS session switches between representations, the uCDN and dCDN would have to agree on things such as:

- o whether the session will be represented by a single log entry (which therefore cannot convey the distribution across representations) or multiple log entries such as one entry per contiguous period at a given representation (which therefore would be generally very difficult to correlate back into a single session)
- o what would the single URI included in the log entry correspond to? the Manifest/top-level-playlist/next-level-playlist,...

The authors feel that since explicit agreement is needed between uCDN and dCDN on how to perform/interpret the summarization, then, to this can only work if specified as part of the CDNI Logging interface and then effectively boils down to Option 4.4 defined below ("Full HAS awareness/per-Session-Logs" Approach).

We note that support by CDNI of a mechanism (independent of HAS) allowing the customization of the fields to be reported in log entries by the dCDN to the uCDN would have a mitigation effect on the HAS logging scaling concerns because it ensures that only the necessary subset of fields are actually stored, reported and processed.

3.4.2.2. Option 4.2: "CDNI Metadata Content Collection ID" Approach

In this approach, a "Content Collection Identifier (CCID)" field is distributed through the CDNI Metadata Interface and the same CCID value is associated through the CDNI Metadata interface with every chunk of the same Content Collection. The CCID value needs to be such that it allows, in combination with the content URI, to uniquely identify a Content Collection. When distributed, and CCID logging is requested from the dCDN, the dCDN Surrogates are to store the CCID value in the corresponding log entries. The objective of this field is to facilitate optional summarization of per-chunk records at step [f] into something along the lines of per-HAS-session logs, at least for the Log consuming applications that do not require per-chunk detailed information (for example billing).

We note that, if the downstream CDN happens to have sufficient HAS awareness to be able to generate a "Session Identifier (Session-ID)", optionally including such Session-ID (in addition to the CCID) in the per-chunk log record would further facilitate optional summarization performed at step [f]. The Session-ID value to be included in a log record by the delivering CDN is such that

- o different per-chunk log records with the same Session-ID value must correspond to the same user session (i.e delivery of same content to same enduser at a given point in time).
- o log records for different chunks of the same user session (i.e delivery of same content to same enduser at a given point in time) should be provided with the same session-ID value. While undesirable, there may be situations where the delivering CDN uses more than one session-ID value for different per-chunk log records of a given session, for example in scenarios of fail-over or load-balancing across multiple Surrogates and where the delivering CDN does not implement mechanisms to synchronize session-IDs across Surrogates.

Effect on CDNI interfaces:

- o CDNI Metadata interface: One additional metadata field (CCID) in CDNI Metadata Interface. We note that a similar Content Collection ID is discussed for handling of other aspects of HAS

and observe that further thought is needed to determine whether such CCID should be shared for multiple purposes or should be independent.

- o CDNI Logging interface: Two additional fields (CCID and Session-ID) in CDNI Logging records.

Advantages/Drawbacks:

- + No information loss (i.e. all details of each individual chunk delivery are preserved). While this full level of detail may not be needed for some Log consuming applications (e.g. billing), this full level of detail is likely valuable (possibly required) for some Log consuming applications (e.g. debugging)
- + Easier integration (at least in the short term) into existing Logging tools since those are all capable of handling per-chunk records
- + Very minor extension to CDNI interfaces needed
- + Facilitated summarization of records related to a HAS session in step [f] and therefore ability to operate on lower volume of logging information in step [g] by log consuming applications that do not need per-chunk record details (e.g. billing) or that need per-session information (e.g. analytics)
- High volume of logging information to be handled (storing & processing) at every step of the Logging process from [a] to [f].

3.4.2.3. Option 4.3: "CDNI Logging Interface Compression" Approach

In this approach, a loss-less compression technique is applied to the sets of Logging records (e.g. Logging files) for transfer on the IETF CDNI Logging Interface. The objective of this approach is to reduce the volume of information to be stored and transferred in step [e].

Effect on CDNI interfaces:

- o One additional compression mechanism to be included in the CDNI Logging Interface

Advantages/Drawbacks:

- + No information loss (i.e. all details of each individual chunk delivery are preserved). While this full level of detail may not be needed for some Log consuming applications (e.g. billing),

this full level of detail is likely valuable (possibly required) for some Log consuming applications (e.g. debugging)

- + Easier integration (at least in the short term) into existing Logging tools since those are all capable of handling per-chunk records
- + Small extension to CDNI interfaces needed
- + Reduced volume of logging information in step [e]
- + Compression likely to be also applicable to logs for non-HAS content
- High volume of logging information to be handled (storing & processing) at every step of the Logging process from [a] to [g], except [e].

3.4.2.4. Option 4.4: "Full HAS awareness/per-Session-Logs" Approach

In this approach, HAS-awareness is assumed across the CDNs interconnected via CDNI and the necessary information to describe the HAS relationship across all chunks of the same Content Collection is distributed through the CDNI Metadata Interface. In this approach, the dCDN Surrogates leverage the HAS information distributed through the CDNI metadata and their HAS-awareness to generate summarized logging information in the very first place (or alternatively, if per-chunk-logs are generated, to accurately correlate and summarize per-chunk-logs into per-session logs) for exchange over the CDNI Logging interface. The objective of that approach is to operate on lower volume of logging information as early as possible in the successive steps of the Logging process.

Effect on CDNI interfaces:

- o CDNI Metadata Interface: significant extension to convey HAS relationship across chunks of a Content Collection. Note that this extension requires specific support for every HAS-protocol to be supported over the CDNI mesh
- o CDNI Logging Interface: extension to specify summarized per-session logs

Advantages/Drawbacks:

- + Lower volume of logging information to be handled (storing & processing) at every step of the Logging process from [a] to [g]

- + Accurate generation of summarized logs because of HAS awareness in dCDN (for example, where the Surrogate is also serving the Manifest File(s) for a content collection, the Surrogate may be able to extract definitive information about the relationship between all chunks)
- Very significant extensions to CDNI interfaces needed including per HAS-protocol specific support
- Very significant additional requirement for HAS awareness on dCDN and for this HAS-awareness to be consistent with the defined CDNI Logging summarization
- Some information loss (i.e. all details of each individual chunk delivery are not preserved). The actual information loss depends on the summarization approach selected (typically the lower the information loss, the lower the summarization gain) so the right sweet-spot would have to be selected. While full level of detail may not be needed for some Log consuming applications (e.g. billing), the full level of detail is likely valuable (possibly required) for some Log consuming applications (e.g. debugging)
- Less easy integration (at least in the short term) into existing Logging tools since those are all capable of handling per-chunk records and may not be capable of handling CDNI summarized records
- Challenges in defining behavior (and achieving summarization gain) in the presence of load-balancing of a given HAS-session across multiple Surrogates (in same or different dCDN)

3.4.3. Recommendation

Because of its benefits (in particular simplicity, universal support by CDNs and support by all log-consuming applications), the authors recommend that the per-chunk logging of Option 4.1 be supported by the CDNI Logging interface as a "High Priority" (as defined in [I-D.draft-ietf-cdni-requirements]) and be a mandatory capability of CDNs implementing CDNI.

Because of its very low complexity and its benefits in facilitating some useful scenarios (e.g. per-session analytics), we recommend that the CCID mechanisms and Session-ID mechanism of Option 4.2 be supported by the CDNI Metadata interface and the CDNI Logging interface as a "Medium Priority" (as defined in [I-D.draft-ietf-cdni-requirements]) and be an optional capability of CDNs implementing CDNI.

The authors also recommend that:

- (i) the ability for the uCDN to request that the CCID and Session-ID field be included in log entries provided by the dCDN be supported by the relevant CDNI interfaces, and
- (ii) the ability for the dCDN to include the CCID field and Session-ID in CDNI log entries (when the dCDN is capable of doing so) and indicate so inside the CDNI Logging interface (in line with the "customizable" log format expected to be defined independently of HAS),

be supported as a "Medium Priority" (as defined in [I-D.draft-ietf-cdni-requirements]) and be an optional capability of CDNs implementing CDNI.

When performing dCDN selection, an uCDN may want to take into account whether a given dCDN is capable of reporting the CCID and Session-ID. Thus, the authors recommend that the ability for a dCDN to advertise its support of the optional CCID and Session-ID capability be supported by the CDNI request Routing /Footprint and Capabilities Advertisement Interface as a "Medium Priority" (as defined in [I-D.draft-ietf-cdni-requirements]).

The authors also recommend that a generic mechanism (independent of HAS) be supported allowing the customization of the fields to be reported in logs by CDNs over the CDNI Logging Interface because of the reduction of the logging information volume exchanged across CDNs by removing the information that is not of interest to the other CDN.

Because it can be achieved with very little complexity and it provides some clear storage/communication compression benefits, the authors recommend that, in line with the concept of Option 4.3, some existing very common compression techniques (e.g. gzip) be supported by the CDNI Logging interface as a "Medium Priority" (as defined in [I-D.draft-ietf-cdni-requirements]) and be an optional capability of CDNs implementing CDNI.

Because of its complexity, the time it would take to understand the trade-offs of candidate summarization approaches and the time it would take to specify the corresponding support in the CDNI Logging interface, the authors recommend that the log summarization discussed in option 4.4 not be supported by the CDNI Logging interface at this stage and be kept as a candidate topic of great interest for a rechartering of the CDNI WG once the first set of deliverables is produced. When doing so, we suggest to investigate the notion of complementing the "push-style" CDNI logging interface supporting summarization by an on-demand pull-type of interface allowing an uCDN to request the subset of the detailed logging information that it may need but is lost in the summarized pushed information.

The authors note that while a CDN only needs to adhere to the CDNI Logging interface on its external interfaces and can perform logging in a different format within the CDN, any possible CDNI Logging approach effectively places some constraints on the dCDN logging format. For example, to support the "Do-Nothing" Approach, a CDN need to perform and retain per chunk logs. As another example, to support the "Full HAS awareness/per-Session-Logs" Approach, the dCDN cannot operate on logging format that summarize "more than" or "in an incompatible way with" the summarization specified for CDNI Logging. However, the authors feel such constraints are (i) inevitable, (ii) outweighed by the benefits of a standardized logging interface and (iii) acceptable because in case of incompatible summarization, all/most CDNs are capable of reverting to per-chunk logging as per the Do-Nothing Approach that we recommend as the base mandatory approach.

3.5. URL Signing

URL Signing is an authorization method for content delivery. This is based on embedding the HTTP URL with information that can be validated to ensure the request has legitimate access to the content. There are two parts: 1) parameters that convey authorization restrictions (e.g. source IP address and time period) and/or protected URL portion, and 2) message digest that confirms the integrity of the URL and authenticates the URL creator. The authorization parameters can be anything agreed upon between the entity that creates the URL and the entity that validates the URL. A key is used to generate the message digest (i.e. sign the URL) and validate the message digest. The two functions may or may not use the same key.

There are two types of keys used for URL Signing: asymmetric keys and symmetric key. Asymmetric keys always have a key pair made up of a public key and private key. The private key and public key are used for signing and validating the URL, respectively. A symmetric key is the same key that is used for both functions. Regardless of the type of key, the entity that validates the URL has to obtain the key. Distribution for the symmetric key requires security to prevent others from taking it. Public key can be distributed freely while private key is kept by the URL signer. The method for key distribution is out of scope.

URL Signing operates in the following way. A signed URL is provided by the content owner (i.e. URL signer) to the user during website navigation. When the user selects the URL, the HTTP request is sent to the CDN which validates that URL before delivering the content.

3.5.1. HAS Implications

The authorization lifetime for URL Signing is affected by HAS. The expiration time in the authorization parameters of URL Signing limits the period that the content referenced by the URL can be accessed. This works for URL that directly access the media content. But for HAS content, the Manifest File contains another layer of URL that reference the chunks. The chunk URL that is embedded in the content may be requested at an indeterminate amount of time later. The time period between access to the Manifest File and chunk retrieval may vary significantly. The type of content (i.e. Live or VoD) impacts the time variance as well. HAS content has this property that needs to be addressed for URL Signing.

3.5.2. CDNI Considerations

For CDNI, the two types of request routing are DNS-based and HTTP-based. The use of symmetric vs. asymmetric key for URL Signing has implications on the trust model between CSP and CDNs and the key distribution method that can be used.

DNS-based request routing does not change the URL. In the case of symmetric key, the CSP and the Authoritative CDN have a business relationship that allows them to share a key (or multiple keys) for URL Signing. When the user request a content from the Authoritative CDN, the URL is signed by the CSP. The Authoritative CDN (as a Upstream CDN) redirects the request to a Downstream CDN via DNS. There may be more than one level of redirection to reach the Delivering CDN. The user would obtain the IP address from DNS and send the HTTP request to the Delivering CDN, which needs to validate the URL. This requires the key to be distributed from Authoritative CDN to the Delivering CDN. This may be problematic when the key is exposed to the Delivering CDN that does not have relationship with the CSP. The combination of DNS-based request routing and symmetric key function is a generic issue for URL Signing and not specific to HAS content. In the case of asymmetric keys, CSP signs URL with its private key. The Delivering CDN validates the URL with the associated public key.

HTTP request routing changes the URL during redirection procedure. In the case of symmetric key, CSP signs the original URL with the same key used by the Authoritative CDN to validate the URL. The Authoritative CDN (as a Upstream CDN) redirects the request to the Downstream CDN. The new URL is signed by the Upstream CDN with the same key used by the Downstream CDN to validate that URL. The key used by the Upstream CDN to validate the original URL is expect to be different than the key used to sign the new URL. In the case of asymmetric keys, CSP signs the original URL with its private key. Authoritative CDN validates that URL with the CSP's public key. The Authoritative CDN redirects the request to the Downstream CDN. The

new URL is signed by the Upstream CDN with its private key. The Downstream CDN validates that URL with the Upstream CDN's public key. There may be more than one level of redirection to reach the Delivering CDN. The URL Signing operation described previously applies at each level between the Upstream CDN and Downstream CDN for both the symmetric key and asymmetric keys cases.

URL Signing requires support in most of the CDNI Interfaces. The CDNI Metadata interface should specify the content that is subject to URL signing and provide information to perform the function. The Downstream CDN should inform the Upstream CDN that it supports URL Signing in the asynchronous capabilities information advertisement as part of the Request Routing interface. This allows the CDN selection function in request routing to choose the Downstream CDN with URL signing capability when the CDNI metadata of the content requires this authorization method. The Logging interface provides information on the authorization method (e.g. URL Signing) and related authorization parameters used for content delivery. Having the information in the URL is not sufficient to know that the surrogate enforced the authorization. URL Signing has no impact on the Control interface.

3.5.3. Option 5.1: Do Nothing

"Do Nothing" approach means that CSP can only perform URL Signing for the top level Manifest File. The top level Manifest File contains chunk URLs or lower level Manifest File URLs, which are not modified (i.e. no URL Signing for the embedded URLs). In essence, the lower level Manifest Files and chunks are delivered without content access authorization.

Effect on CDNI interfaces:

- o None

Advantages/Drawbacks:

- + Top level Manifest File access is protected
- + Upstream CDN and Downstream CDN do not need to be aware of HAS content
- Lower level Manifest Files and chunks are not protected, making this approach unqualified for content access authorization

3.5.4. Option 5.2: Flexible URL Signing by CSP

In addition to URL Signing for the top level Manifest File, CSP performs flexible URL Signing for the lower level Manifest Files and chunks. For each HAS session, the top level Manifest File contains signed chunk URLs or signed lower level Manifest File URLs for the specific session. The lower level Manifest File contains session-based signed chunk URLs. CSP generates the Manifest Files dynamically for the session. The chunk (segment/fragment) is delivered with content access authorization using flexible URL Signing which protects the invariant portion of the URL. Segment URL (e.g. HLS) is individually signed for the invariant URL portion (Relative URL) or the entire URL (Absolute URL without Redirection) in the Manifest File. Fragment URL (e.g. Smooth Streaming) is signed for the invariant portion of the template URL in the Manifest File. More details are provided later in this section. The URL Signing expiration time for the chunk needs to be long enough to play the video. There are implications of signing the URLs in the Manifest File. For Live content, the Manifest Files are requested at a high frequency. For VoD content, the Manifest File may be quite large. URL Signing can add more computational load and delivery latency in high volume cases.

For HAS content, the Manifest File contains the Relative Locator, Absolute Locator without Redirection, or Absolute Locator with Redirection for specifying the chunk location. Signing the chunk URL requires CSP to know the portion of the URL that remains when the content is requested from the Delivery CDN surrogate.

For Absolute URL without Redirection, the CSP knows that the chunk URL which is explicitly linked with the delivery CDN surrogate and can sign the URL based on that information. Since the entire URL is set and does not change, the surrogate can validate the URL. The CSP and the Delivery CDN are expected to have a business relationship in this case. So either symmetric key or asymmetric keys can be used for URL Signing.

For Relative URL, the URL of the Manifest File provides the root location. The method of request routing affects the URL used to ultimately request the chunk from the Delivery CDN surrogate. For DNS, the original URL does not change. This allows CSP to sign the chunk URL based on the Manifest File URL and the Relative URL. For HTTP, the URL changes during redirection. In this case, CSP does not know the redirected URL that will be used to request the Manifest File. This uncertainty makes it impossible to accurately sign the chunk URLs in the Manifest File. Basically, URL Signing using this reference method, "as is" for entire URL protection, is not supported. However, instead of signing the entire URL, the CSP signs the Relative URL (i.e. invariant portion of the URL) and conveys the protected portion in the authorization parameters embedded in the

chunk URL. This approach works the same way as Absolute URL without Redirection, except the HOST part and (part of) the PATH part of the URL are not signed and validated. The security level should remain the same as content access authorization ensures that the user that requested the content has the credentials. This scheme does not seem to compromise the authorization model since the resource is still protected by the authorization parameters and message digest. Perhaps, further evaluation on security would be helpful.

For Absolute URL with Redirection, the method of request routing affects the URL used to ultimately request the chunk from the Delivery CDN surrogate. This case has the same conditions as the Relative URL. The difference is that the URL is for the chunk instead of the Manifest File. For DNS, the chunk URL does not change and can be signed by the CSP. For HTTP, the URL used to deliver the chunk is unknown to the CSP. In this case, CSP cannot sign the URL and this method of reference for the chunk is not supported.

Effect on CDNI interfaces:

- o Requires the ability to exclude the variant portion of URL in the signing process (NOTE: Issue is specific to URL Signing support for HAS content and not CDNI?)

Advantages/Drawbacks:

- + Manifest File and chunks are protected
- + Upstream CDN and Downstream CDN do not need to be aware of HAS content
- + DNS-based request routing with asymmetric keys and HTTP-based request routing for Relative URL and Absolute URL without Redirection works
- CSP has to generate Manifest Files with session-based signed URLs and becomes involved in content access authorization for every HAS session
- Manifest Files are not cacheable
- DNS-based request routing with symmetric key may be problematic due to need for transitive trust between CSP and Delivery CDN
- HTTP-based request routing for Absolute URL with Redirection does not work because the URL used Delivery CDN surrogate is unknown to the CSP

3.5.5. Option 5.3: Flexible URL Signing by Upstream CDN

This is similar to the previous section, with the exception that the Upstream CDN performs flexible URL for the lower level Manifest Files and chunks. URL Signing for the top level Manifest File is still provided by the CSP.

Effect on CDNI interfaces:

- o Requires the ability to exclude the variant portion of URL in the signing process (NOTE: Issue is specific to URL Signing support for HAS content and not CDNI?)

Advantages/Drawbacks:

- + Manifest File and chunks are protected
- + CSP does not need to be involved in content access authorization for every HAS session
- + Downstream CDN does not need to be aware of HAS content
- + DNS-based request routing with asymmetric keys and HTTP-based request routing for Relative URL and Absolute URL without Redirection works
- Upstream CDN has to generate Manifest Files with session-based signed URLs and becomes involved in content access authorization for every HAS session
- Manifest Files are not cacheable
- Manifest File needs to be distributed through the uCDN
- DNS-based request routing with symmetric key may be problematic due to need for transitive trust between uCDN and non-adjacent Delivery CDN
- HTTP-based request routing for Absolute URL with Redirection does not work because the URL used Delivery CDN surrogate is unknown to the uCDN

3.5.6. Option 5.4: Authorization Group ID and HTTP Cookie

Based on the Authorization Group ID metadata, CDN validates the URL Signing or validates the HTTP cookie for request of content in the group. CSP performs URL Signing for the top level Manifest File. The top level Manifest File contains lower level Manifest File URLs

or chunk URLs. The lower level Manifest Files and chunks are delivered with content access authorization using HTTP cookie that contains session state associated with authorization of the top level Manifest File. The Group ID Metadata is used to associate the related content (i.e. Manifest Files and chunks). It also specifies content (e.g. regexp method) that needs to be validated by either URL Signing or HTTP cookie. Note that the creator of the metadata is HAS-aware. Duration of the chunk access may be included in the URL Signing of the top level Manifest File and set in the cookie. Alternatively, the access control duration could be provided by the CDNI Metadata interface.

Effect on CDNI interfaces:

- o CDNI Metadata Interface - Authorization Group ID metadata identifies the content that is subject to validation of URL Signing or validation of HTTP cookie associated with the URL Signing
- o CDNI Logging Interface - Report the authorization method used to validate the request for content delivery

Advantages/Drawbacks:

- + Manifest File and chunks are protected
- + CDN does not need to be aware of HAS content
- + CSP does not need to change the Manifest Files
- Authorization Group ID metadata is required (i.e. CDNI Metadata Interface enhancement)
- Requires the use of HTTP cookie which may not be acceptable in some environments (e.g. where some targeted User-Agents do not support HTTP Cookie)
- Manifest File has to be delivered by surrogate

3.5.7. Option 5.5: HAS-awareness with HTTP Cookie in CDN

CDN is aware of HAS content and uses URL Signing and HTTP cookie for content access authorization. URL Signing is fundamentally about authorizing access to a Content Item or its specific Content Collections (representations) for a specific user during a time period with possibly some other criteria. A chunk is an instance of the sets of chunks referenced by the Manifest File for the Content Item or its specific Content Collections. This relationship means

that once the Downstream CDN has authorized the Manifest File, it can assume that the associated chunks are implicitly authorized. The new function for the CDN is to link the Manifest File with the chunks for the HTTP session. This can be accomplished by using an HTTP cookie for the HAS session.

After validating the URL and detecting that the requested content is a top level Manifest File, the delivery CDN surrogate sets a HTTP cookie with a signed session token for the HTTP session. When a request for a lower level Manifest File or chunk arrives, the surrogate confirms that the HTTP cookie value contains the correct session token. If so, the lower level Manifest File or chunk is delivered due to transitive authorization property. Duration of the chunk access may be included in the URL Signing of the top level Manifest File and set in the cookie. The details of the operation are left to be determined later.

Effect on CDNI interfaces:

- o CDNI Metadata Interface - New metadata identifies the content that is subject to validation of URL Signing and information in the cookie for the type of HAS content
- o Request Routing interface - Downstream CDN should inform the Upstream CDN that it supports URL Signing for known HAS content types in the asynchronous capabilities information advertisement. This allows the CDN selection function in request routing to choose the appropriate Downstream CDN when the CDNI metadata identifies the content
- o CDNI Logging Interface - Report the authorization method used to validate the request for content delivery

Advantages/Drawbacks:

- + Manifest File and chunks are protected
- + CSP does not need to change the Manifest Files
- Requires full HAS awareness on part of Upstream CDN and Downstream CDN
- Requires CDNI Interfaces extensions
- Requires the use of HTTP cookie which may not be acceptable in some environments (e.g. where some targeted User-Agents do not support HTTP Cookie)

- Manifest File has to be delivered by surrogate

3.5.8. Option 5.6: HAS-awareness with Manifest in CDN

CDN is aware of HAS content and uses URL Signing for content access authorization of Manifest File and chunk. CDN generates or rewrites the Manifest Files and learns about the chunks based on the Manifest File. The embedded URLs in the Manifest File are signed by the CDN. Duration of the chunk access may be included in the URL Signing. The details of the operation are left to be determined later. Since this approach is based on signing the URLs in the Manifest File, the implications for Live and VoD content mentioned in Section 3.5.4 apply.

Effect on CDNI interfaces:

- o CDNI Metadata Interface - New metadata identifies the content that is subject to validation of URL Signing and information in the cookie for the type of HAS content
- o Request Routing interface - Downstream CDN should inform the Upstream CDN that it supports URL Signing for known HAS content types in the asynchronous capabilities information advertisement. This allows the CDN selection function in request routing to choose the appropriate Downstream CDN when the CDNI metadata identifies the content
- o CDNI Logging Interface - Report the authorization method used to validate the request for content delivery

Advantages/Drawbacks:

- + Manifest File and chunks are protected
- + CSP does not need to change the Manifest Files
- Requires full HAS awareness on part of Upstream CDN and Downstream CDN
- Requires CDNI Interfaces extensions
- Requires CDN to generate or rewrite the Manifest File
- Manifest File has to be delivered by surrogate

3.5.9. Recommendation

The authors consider that Option 5.1 (Do Nothing) is not suitable for access control of HAS content.

Where the HTTP Cookie mechanism is supported by the targeted User-Agents and the security requirements can be addressed through proper use of HTTP Cookies, the authors recommend use of Option 5.4 (Authorization Group ID with HTTP Cookie) and therefore be supported by the CDNI solution. This method does not require manifest file manipulation which may be a significant deployment obstacle. Otherwise, the authors recommend that Option 5.2 (Flexible URL Signing by the CSP) or Option 5.3 (Flexible URI Signing by the Upstream CDN) be used and therefore that flexible URI be supported by the CDNI solution. Option 5.2 and Option 5.3 protect all the content, does not require Downstream CDN to be aware of HAS, does not impact CDNI interfaces, supports all different types of devices, and supports the common cases of request routing for HAS content (i.e. DNS-based request routing with asymmetric keys and HTTP-based request routing for Relative URL).

HAS-awareness in CDN (Option 5.5 and Option 5.6) have some advantages that should be considered for future support (e.g. CDN that is aware of HAS content can manage the content more efficiently at a broader context. Content distribution, storage, delivery, deletion, access authorization, etc. can all benefit.). Including HAS-awareness as part of the current CDNI charter, however, would almost certainly delay the CDNI WG's milestones, and the authors therefore do not recommend it right now.

3.6. Content Purge

At some point in time, a uCDN might want to remove content from a dCDN. With regular content, this process can be relatively straightforward; a uCDN will typically send the request for content removal to the dCDN including a reference to the content which it wants to remove (e.g. in the form of a URL). Due to the fact that HAS content consists of large groups of files however, things might be more complex. Section 3.1 describes a number of different scenarios for doing file management on these groups of files, while Section 3.2 list the options for performing Content Acquisition on these Content Collections. This section will present the options for requesting a Content Purge for the removal of a Content Collection from a dCDN.

3.6.1. Option 6.1: No HAS awareness

The most straightforward way to signal content purge requests is to just send a single purge request for every file that makes up the Content Collection. While this method is very simple and does not require HAS awareness, it obviously creates a signaling overhead between the uCDN and dCDN since a reference is to be provided for each content chunk to be purged.

Effect on CDNI interfaces:

- o None

Advantages/Drawbacks (apart from those listed under Option 3.3):

- + Does not require changes to the CDNI Interfaces or HAS awareness
- Requires individual purge request for every file making up a Content Collection (or, alternatively, requires the ability to convey references to all the chunks making up a Content Collection inside a purge request) which creates signaling overhead

3.6.2. Option 6.2: Purge Identifiers

There exists a potentially more efficient method for performing content removal of large numbers of files simultaneously. By including a "Purge Identifier (Purge-ID)" in the metadata of a particular file, it is possible to virtually group together different files making up a Content Collection. A Purge-ID can take the form of an arbitrary number or string which is communicated as part of the CDNI Metadata Interface and which is the same for all files making up a particular Content Item, and different across different Content Items. If a uCDN wants to request the dCDN to remove a Content Collection, it can send a purge request containing this Purge-ID. The dCDN can then remove all files that share the corresponding Purge-ID.

The advantage of this method is that it is relatively simple to use by both the dCDN and uCDN and requiring only limited additions to the CDNI Metadata Interface and CDNI Control Interface.

The Purge-ID is similar to the Content Collection ID discussed in Section 3.4.2.2 for handling HAS Logging and we note that further thought is needed to determine whether the CCID and Purge-ID should be collapsed into a single element or remain separate elements.

Effect on CDNI interfaces:

- o CDNI Metadata Interface: Add metadata field for indicating Purge-ID
- o CDNI Control Interface: Add functionality to convey a Purge-ID in purge requests

Advantages/Drawbacks:

- + Allows for efficient purging of content from a dCDN
- + Does not require HAS awareness on part of dCDN

3.6.3. Recommendation

Based on the listed pros and cons, the authors recommend the WG to have mandatory support Option 1.1, the 'Do Nothing'-approach. In addition, because of its very low complexity and its benefit in facilitating low-overhead purge of large numbers of content items simultaneously, the authors recommend that the Purge IDentifier of Option 6.2 be supported as an optional feature by the CDNI Metadata interface and the CDNI Control interface.

3.7. Other issues

This section includes some HAS-specific issues that came up during the discussion of this draft and which do not fall under any of the categories discussed in the previous sections.

- As described in Section 2.2, a Manifest File might either be delivered by a CDN or by the CSP, thereby being invisible to the CDN delivering the chunks. Obviously, the decision on whether the CDN or CSP delivers the Manifest File is made between the uCDN and CSP, and the dCDN has no choice in the matter. However, some dCDNs might only want to offer their services in the cases where they have access to the Manifest File (e.g. because their internal architecture is based around the knowledge inside the Manifest File). For these cases, it might be useful to include a field in the CDNI Capability Advertisement to allow dCDNs to advertise the fact that they require access to the Manifest File.

4. IANA Considerations

This document makes no request of IANA.

5. Security Considerations

This document does not discuss security issues around HTTP or HAS delivery. Those are expected to be discussed in the CDNI WG documents including [I-D.ietf-cdni-framework].

6. Acknowledgements

The authors would like to thank Kevin Ma, Stef van der Ziel, Bhaskar Bhupalam, Mahesh Viveganandhan, Larry Peterson, Ben Niven-Jenkins and Matt Caulfield for their valuable contributions to this document.

7. References

7.1. Normative References

- [RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", RFC 6707, September 2012.

7.2. Informative References

- [I-D.draft-bertrand-cdni-logging]
Bertrand, G., Ed. and E. Stephan, "CDNI Logging Interface", .
- [I-D.draft-ietf-cdni-requirements]
Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements, draft-ietf-cdni-requirements-03", June 2012.
- [I-D.ietf-cdni-framework]
Peterson, L. and B. Davie, "Framework for CDN Interconnection", draft-ietf-cdni-framework-03 (work in progress), February 2013.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax, RFC3986", January 2005.

Authors' Addresses

Ray van Brandenburg
TNO
Brassersplein 2
Delft 2612CT
the Netherlands

Phone: +31-88-866-7000
Email: ray.vanbrandenburg@tno.nl

Oskar van Deventer
TNO
Brassersplein 2
Delft 2612CT
the Netherlands

Phone: +31-88-866-7000
Email: oskar.vandeventer@tno.nl

Francois Le Faucheur
Cisco Systems
Greenside, 400 Avenue de Roumanille
Sophia Antipolis 06410
France

Phone: +33 4 97 23 26 19
Email: flefauch@cisco.com

Kent Leung
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: +1 408-526-5030
Email: kleung@cisco.com

INTERNET-DRAFT
Intended Status: Standards Track
Expires: January 5, 2013

TS Choi
ETRI
YI Seo
DJ Kim
KT
JM Lee
SKT
JR Koo
LGU+
JDH Shinn
Solbox Inc.
KS Park
KAIST
July 4, 2012

CDNi Request Routing Redirection with Loop Prevention
draft-choi-cdni-req-routing-redir-loop-prevention-00

Abstract

This document describes request routing redirection procedures, loop prevention mechanisms, and other operational considerations which are associated with redirection.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 29, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1	Introduction	3
1.1	Terminology	3
2.	Redirection Procedures	3
2.1.	Preliminaries	3
2.2.	Iterative Redirection Procedures	4
2.2.1.	HTTP-based Redirection	4
2.2.2.	DNS-based Redirection	8
2.3.	Recursive Redirection Procedures	10
2.3.1.	HTTP-based Redirection	10
2.3.2.	DNS-based Redirection	14
3.	Redirection Loop Prevention	17
4.	Redirection Operational Considerations	18
5	Security Considerations	19
6	IANA Considerations	19
7	References	19
7.1	Normative References	19
7.2	Informative References	19
	Authors' Addresses	19

1 Introduction

According to the CDNi generic and request routing interface requirements[I-D.ietf-cdni-requirements], the CDNi solution shall support iterative and recursive CDNi request routing, efficient request routing for small and large objects, arbitrary number of levels of cascaded CDN redirection, looping prevention of any CDN request routing redirection, and subsequently allowing the request routing redirection. To meet such requirements, this document describes request routing redirection procedures, loop prevention mechanisms, and other operational considerations that are associated with redirection.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Redirection Procedures

2.1. Preliminaries

To meet the requirements of request routing redirection, we define the term "CDN-Provider-ID". It uniquely identifies each CDN provider during the course of request routing redirection. It consists of "CDN provider list" and "MaxNumRedHops". The CDN provider list is a set of uniquely identifiable CDN provider names. A pair of an AS number and an additional qualifier is used for CDN provider name. MaxNumRedHops represents a maximum allowed redirections. The value is decreased once every redirection occurs until it reaches 0. To avoid its usage abuse (end user or CDN operator can set huge number like 100 or above), a reasonable upper bound has to be agreed among CDN providers. Security aspect of it is for further study.

Since more than one CDN providers can belong to the same AS, an additional qualifier is used to guarantee the uniqueness. A few examples of CDN provider name are 100:0 and 200:1. The former means that a CDN provider belong to AS 100 and it is the only CDN provider within that AS. The latter represents the first CDN provider in the AS 200. There are other CDN providers in the same AS.

One example of CDN-Provider-ID is 100:0:10, which means that a CDN provider that belong to AS number 100 and it is the only CDN provider and a maximum allowed redirection is 10.

It is applicable for both HTTP-based and DNS redirections. For HTTP-

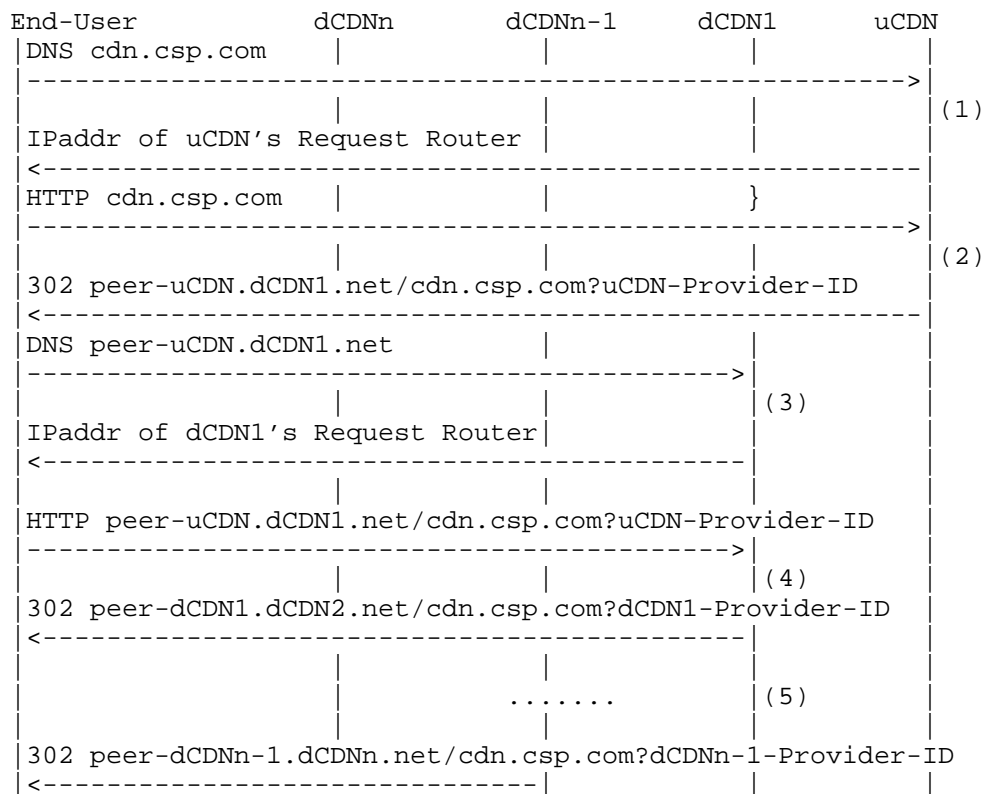
based redirection, we define a HTTP request routing redirection header "CDN-Provider-ID". For each step of redirection, it is attached to the beginning of the provider domain URL. For example, uCDN initiates a redirection with its URL, `http://100:0:10.cdn.csp.com`. dCDN further attaches its own CDN-Provider-ID in the front when another level of redirection is required. For DNS-based redirection, the CDN-Provider-ID can be attached in the DNS CNAME.

Since the CDNi requirements for the support of arbitrary topology of interconnected CDNs, this document assumes that the redirection procedures and loop prevention mechanisms support arbitrary topology.

2.2. Iterative Redirection Procedures

2.2.1. HTTP-based Redirection

In this section, we describe an iterative procedure of HTTP-based request routing redirection with loop prevention.



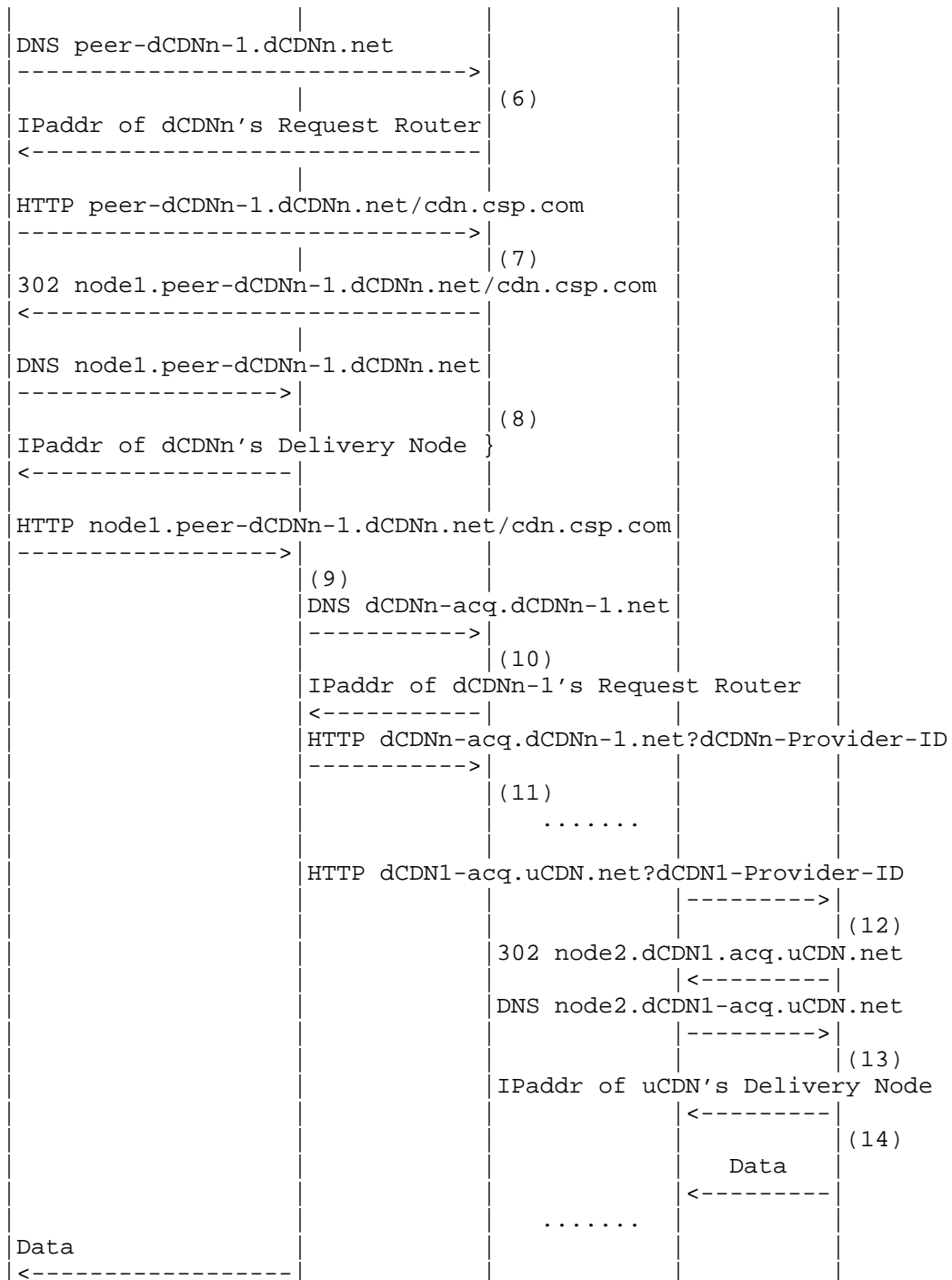


Figure 1: HTTP-based request routing redirection iterative procedure

The steps illustrated in the figure are as follows:

1. A DNS resolver for uCDN provider processes the DNS request for its customer based on CDN-domain `cdn.csp.com`. It returns the IP address of a request router in uCDN provider.
2. A Request Router for uCDN provider processes the HTTP request and recognizes that the end-user is best served by dCDN1. So it returns a 302 redirect message for a new URL constructed by "stacking" dCDN1's distinguished CDN-domain (`peer-uCDN.dCDN1.net`) on the front of the original URL. It also adds uCDN's CDN-Provider-ID in the HTTP request message. It consists of `ASNum:Qualifier:MaxNumRedHops`. This information is not processed by the customer but conveyed in the HTTP message without any modification of the step 4. The details on how it is used for loop prevention is described in the step 4.
3. The end-user does a DNS lookup using dCDN1's distinguished CDN-domain (`peer-uCDN.dCDN1.net`). dCDN1's DNS resolver returns the IP address of a request router for dCDN1.
4. The request router for dCDN1 processes the HTTP request. There are two options: redirect further to another dCDN (i.e., cascading the request) or process it by itself. In either cases, it performs loop prevention step first. It checks `CDN-provider-ID`: the provider list contains a list of CDN providers which requested redirections so far. If either it contains own CDN provider name or `MaxNumRedHops` becomes 0, it means that the redirection loop has occurred or the number of redirection hops has reached the maximum. Once loop is detected, details on the next steps is described in the section 3. If it is loop free, it either redirects further or processes based on the local policy. For the former, it selects another dCDN provider and sends an HTTP redirect message with its own `CDN-Provider-ID` attached. For the latter, it selects a suitable delivery node to serve the end-user request, and returns a 302 redirect message for a new URL constructed by replacing the hostname by a subdomain of the dCDN1's distinguished CDN-domain that points to the selected delivery node. Then it goes to the step 6.
5. If further redirection is decided, it repeats steps 2 - 4 until it either selects dCDN provider to serve the request or `MaxNumRedHops` expires. If the former occurs, it resumes the step 6. If the latter occurs, it follows the processes described in the section 3.

6. Assuming that dCDNn is selected as a serving dCDN provider, the end-user does a DNS lookup using dCDNn's distinguished CDN-domain (peer-dCDNn-1.dCDNn.net). dCDNn-1's DNS resolver returns the IP address of a request router for dCDNn.
7. The request router for dCDN1 processes the HTTP request and selects a suitable delivery node to serve the end-user request, and returns a 302 redirect message for a new URL constructed by replacing the hostname by a subdomain of the dCDNn's distinguished CDN-domain that points to the selected delivery node.
8. The end-user does a DNS lookup using dCDNn's delivery node subdomain (node1.peer-dCDNn-1.dCDNn.net). dCDNn's DNS resolver returns the IP address of the delivery node.
9. The end-user requests the content from dCDNn's delivery node. In the case of a cache hit, steps 10 ~ 14 below do not happen, and the content data is directly returned by the delivery node to the end-user. In the case of a cache miss, the content needs to be acquired by dCDN from either parent dCDN or uCDN (not the CSP). The distinguished CDN-domain peer-dCDNn-1.dCDNn.net indicates that this content is to be acquired from dCDNn-1; stripping the CDN-domain reveals the original CDN-domain cdn.csp.com and dCDNn may verify that this CDN-domain belongs to a known peer (so as to avoid being tricked into serving as an open proxy). It then does a DNS request for an inter-CDN acquisition CDN-domain as agreed above (in this case, dCDNn-acq.dCDNn-1.net). This process repeats recursively until it finds a CDN provider that can serve the requested content.
10. dCDNn-1's DNS resolver processes the DNS request and returns the IP address of a request router in dCDNn-1.
11. The request router for dCDNn-1 processes the HTTP request from dCDNn's delivery node. dCDNn-1 request router recognizes that the request is from a peer CDN rather than an end-user because of the dedicated inter-CDN acquisition domain (dCDNn-acq.dCDNn-1.net). It also performs loop prevention process as described in step 4 based on the provided CDN-Provider-ID. Depending on the number of levels of redirection and availability of contents, the same process repeats until either content serving CDN provider is found or MaxNumRedHps expires.
12. Assuming that all intermediate dCDNs also have a cache miss, The request router for uCDN selects a suitable delivery node to serve the inter-CDN acquisition request and returns a 302

redirect message for a new URL constructed by replacing the hostname by a subdomain of the uCDN's distinguished inter-CDN acquisition domain that points to the selected delivery node.

13. uCDN DNS resolver processes the DNS request and returns the IP address of the delivery node in uCDN.
14. uCDN serves content for the requested CDN-domain to dCDN and finally to end-user. Although not shown, it is at this point that uCDN processes the rest of the URL: it extracts information identifying the origin server, validates that this server has been registered, and determines the content provider that owns the origin server. It may also perform its own content acquisition steps if needed before returning the content to dCDN.

2.2.2. DNS-based Redirection

In this section, we describe an iterative procedure of DNS-based request routing redirection with loop prevention.

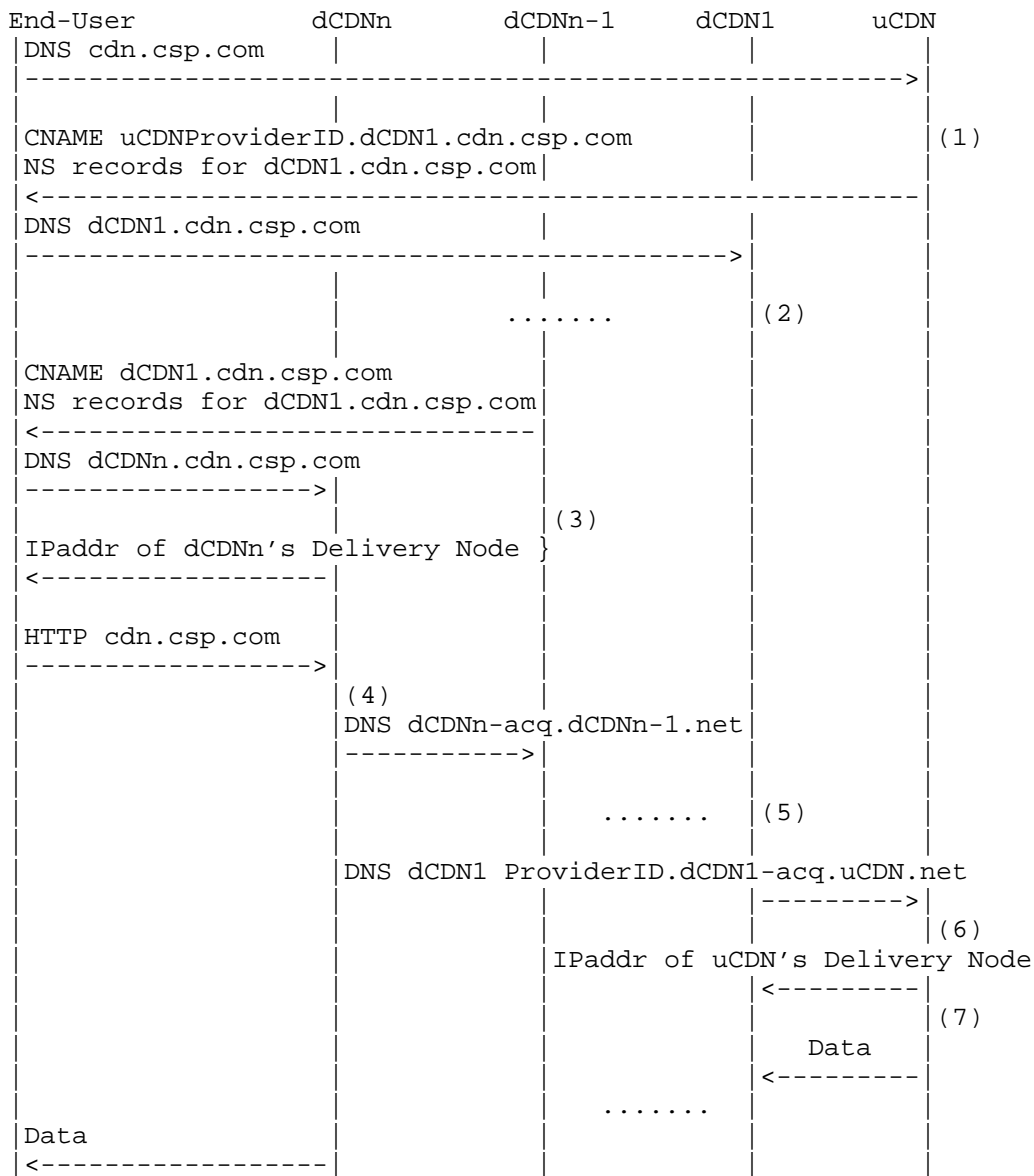


Figure 2: DNS-based request routing redirection iterative procedure

The steps illustrated in the figure are as follows:

1. Request Router for uCDN provider processes the DNS request for CDN- domain `cdn.csp.com` and recognizes that the end-user is best

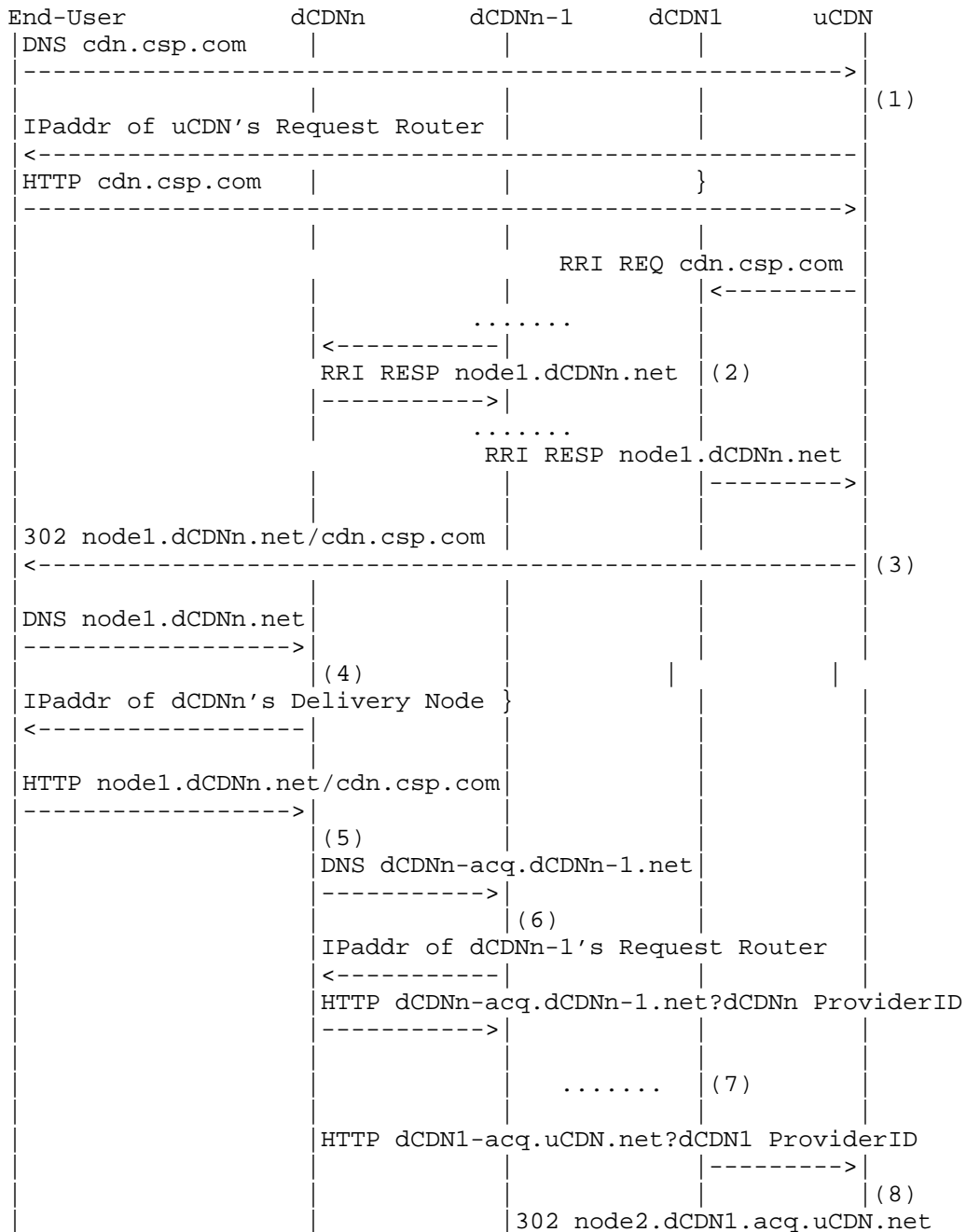
served by another CDN. (This may depend on the IP address of the user's local DNS resolver, or other information discussed below.) The Request Router returns a DNS CNAME response by "stacking" the distinguished identifier for dCDN1 and uCDN's CDN-Provider-ID onto the original CDN-domain (e.g., dCDN1.cdn.csp.com), plus an NS record that maps dCDN1.cdn.csp.com to dCDN1's Request Router.

2. The end-user does a DNS lookup using the modified CDN-domain (i.e., dCDN1.cdn.csp.com). dCDN1 Request Router processes the request and decides to serve the request or redirect further to another CDN provider. It also checks redirection loop. This process iterates until either serving dCDN is selected or MaxNumRedHops expires. In this case, dCDNn is selected as a serving dCDN. If the former occurs, it proceeds to step 3. If the latter occurs, it follows the processes described in the section 3.
3. The end-user does a DNS lookup using the modified CDN-domain (i.e., dCDN1.cdn.csp.com). This causes dCDNn's request router returns an IP address of a suitable delivery node.
4. The end-user requests the content from dCDNn's delivery node. In the case of a cache hit, steps 5 ~ 7 do not happen, and the content data is directly returned by the delivery node to the end-user. In the case of a cache miss, the content needs to be acquired by dCDNn from either parent dCDN or uCDN (not the CSP). It also performs loop prevention process as described in the step 2 based on the provided CDN-Provider-ID.
5. Depending on the number of levels of redirection and availability of contents, the same process repeats until either content serving CDN provider is found or MaxNumRedHops expires.
6. Assuming that all intermediate dCDNs also miss cache, uCDN is selected as a content delivery CDN provider. Thus, the request router for uCDN selects a suitable delivery node to serve the inter-CDN acquisition request and returns IP address of the suitable uCDN delivery node.
7. uCDN serves content to dCDN1 and further down to end-user. Although not shown, it is at this point that uCDN processes the rest of the URL: it extracts information identifying the origin server, validates that this server has been registered, and determines the content provider that owns the origin server.

2.3. Recursive Redirection Procedures

2.3.1. HTTP-based Redirection

In this section, we describe an recursive procedure of HTTP-based request routing redirection with loop prevention.



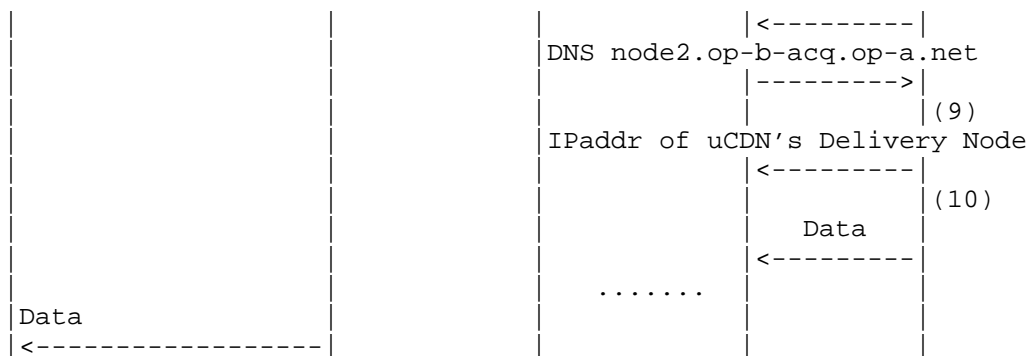


Figure 3: HTTP-based request routing redirection recursive procedure

The steps illustrated in the figure are as follows:

1. A DNS resolver for uCDN provider processes the DNS request for its customer based on CDN-domain `cdn.csp.com`. It returns the IP address of a request router in uCDN provider.
2. A Request Router for uCDN provider processes the HTTP request and recognizes that the end-user is best served by dCDN1. So it queries the CDNI Request Routing interface of dCDN1 providing a set of information about the request including the URL requested. It also provides uCDN's CDN-Provider-ID for loop prevention process. It consists of `ASNum:Qualifier:MaxNumRedHops`. It checks `CDN-Provider-ID`: the provider list contains a list of CDN providers that have requested redirections so far. If either it contains its own CDN provider name or `MaxNumRedHops` becomes 0, it means that the redirection loop has occurred or it has reached the maximum number of allowed number of redirection hops. Once loop is detected, details on the next steps are described in the section 3. If it is loop free, dCDN1 then either replies with the DNS name of a delivery node or redirect to another dCDN. Such cascading redirection can continue until a serving dCDN is decided. The RRI RESP can be sent in the reverse order of cascaded redirection or directly to the redirection origin CDN provider if contact information is known. The contact information can be embedded in the RRI REQ message or pre-configured during bootstrapping process. The default behavior is recursive RESP.
3. uCDN returns a 302 redirect message for a new URL obtained from the Request Routing Interface.

4. The end-user does a DNS lookup using the host name of the URL just provided (node1.dCDNn.net). dCDNn's DNS resolver returns the IP address of the corresponding delivery node. Note that, since the name of the delivery node was already obtained from dCDNn using the CDNI Request Routing Interface, there should not be any further redirection here (in contrast to the iterative method described above.)
4. The request router for dCDN1 processes the HTTP request. There are two options: redirect further to another dCDN (i.e., cascading the request) or process it by itself. In either cases, it performs loop prevention step first. If it is loop free, it either redirect further or processes based on the local policy. For the former, it selects another dCDN provider and sends HTTP redirect message with its own CDN-Provider-ID attached. For the latter, it selects a suitable delivery node to serve the end-user request, and returns a 302 redirect message for a new URL constructed by replacing the hostname by a subdomain of the dCDN1's distinguished CDN-domain that points to the selected delivery node. Then it goes to step 6.
5. The end-user requests the content from dCDNn's delivery node. In the case of a cache hit, steps 6 ~ 10 below do not happen, and the content data is directly returned by the delivery node to the end-user. In the case of a cache miss, the content needs to be acquired by dCDN from either parent dCDN or uCDN (not the CSP). The distinguished CDN-domain dCDNn.net indicates that this content is to be acquired from dCDNn-1; stripping the CDN-domain reveals the original CDN-domain cdn.csp.com and dCDNn may verify that this CDN-domain belongs to a known peer (so as to avoid being tricked into serving as an open proxy). It then does a DNS request for an inter-CDN acquisition CDN-domain as agreed above (in this case, dCDNn-acq.dCDNn-1.net). This process repeats recursively until it finds CDN provider that can serve the requested content.
6. dCDNn-1's DNS resolver processes the DNS request and returns the IP address of a request router in dCDNn-1.
7. The request router for dCDNn-1 processes the HTTP request from dCDNn's delivery node. dCDNn-1 request router recognizes that the request is from a peer CDN rather than an end-user because of the dedicated inter-CDN acquisition domain (dCDNn-acq.dCDNn-1.net). It also performs loop prevention process as described in the step 2 based on the provided CDN-Provider-ID. Depending on the number of levels of redirection and availability of contents, the same process repeats until

either content serving CDN provider is found or MaxNumRedHps expires.

8. Assuming that all intermediate dCDNs also have a cache miss, The request router for uCDN selects a suitable delivery node to serve the inter-CDN acquisition request and returns a 302 redirect message for a new URL constructed by replacing the hostname by a subdomain of the uCDN's distinguished inter-CDN acquisition domain that points to the selected delivery node.
9. uCDN DNS resolver processes the DNS request and returns the IP address of the delivery node in uCDN.
10. uCDN serves content for the requested CDN-domain to dCDN and finally to end-user. Although not shown, it is at this point that uCDN processes the rest of the URL: it extracts information identifying the origin server, validates that this server has been registered, and determines the content provider that owns the origin server. It may also perform its own content acquisition steps if needed before returning the content to dCDN.

2.3.2. DNS-based Redirection

In this section, we describe an recursive procedure of DNS-based request routing redirection with loop prevention.

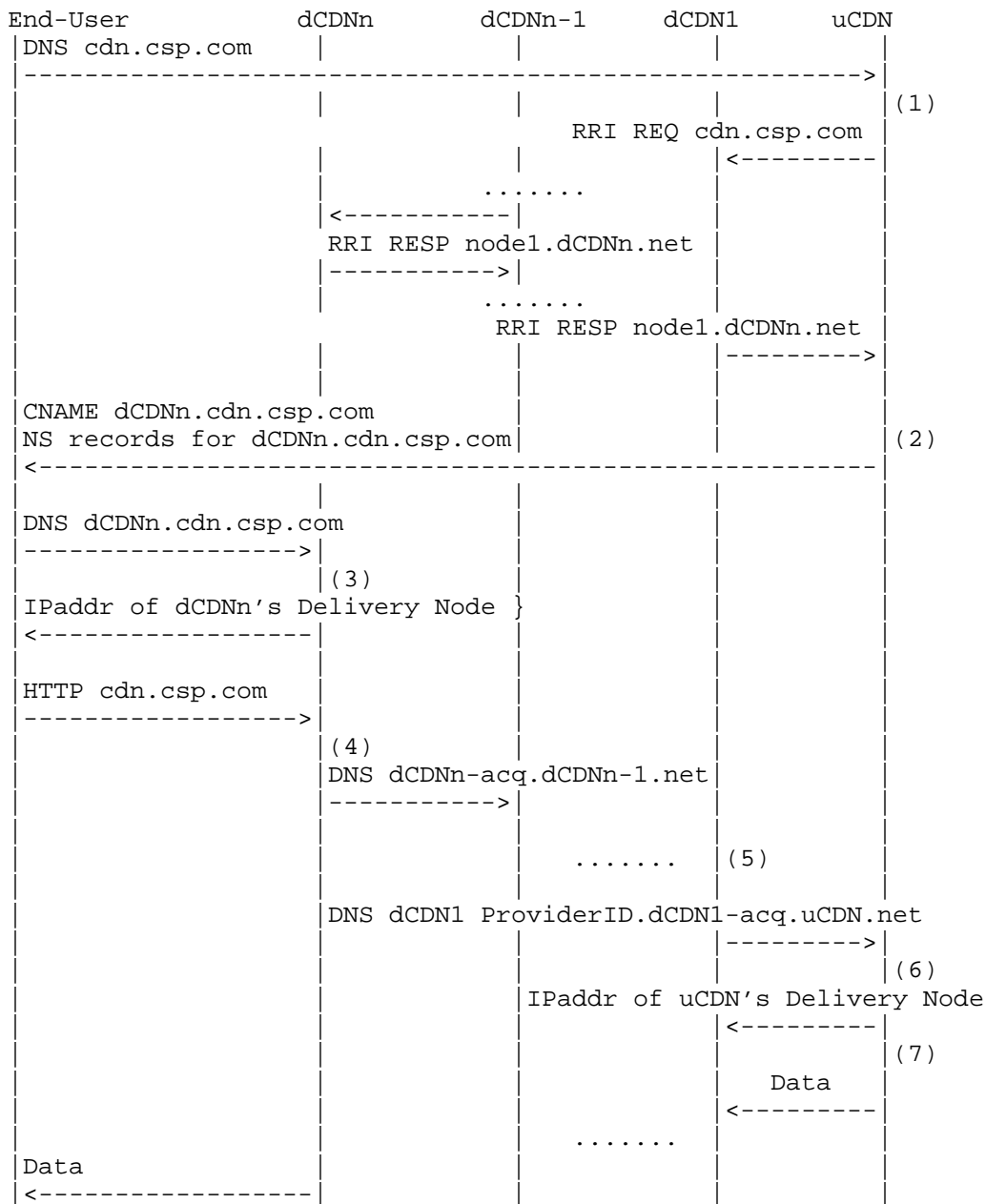


Figure 4: DNS-based request routing redirection recursive procedure

The steps illustrated in the figure are as follows:

1. Request Router for uCDN provider processes the DNS request for CDN-domain `cdn.csp.com` and recognizes that the end-user is best served by dCDN1. So it queries the CDNI Request Routing Interface of dCDN1 providing a set of information about the request including the URL requested. It also provides uCDN's CDN-Provider-ID for loop prevention process. It checks CDN-Provider-ID. If either it contains own CDN provider name or `MaxNumRedHops` becomes 0, it means that the redirection loop is occurred. Once loop is detected, details on the next steps are described in the section 3. If it is loop-free, dCDN1 then either replies with the DNS name of a delivery node or redirect to another dCDN. Such cascaded redirection can continue until a serving dCDN is decided. The RRI RESP can be sent in the reverse order of cascaded redirection or directly to the redirection origin CDN provider if contact information is known. The contact information can be embedded in the RRI REQ message or pre-configured during bootstrapping process. The default behavior is recursive RESP. In this case, dCDNn is selected as a serving dCDN.
2. The Request Router returns a DNS CNAME response by "stacking" the distinguished identifier for dCDNn and uCDN's CDN-Provider-ID onto the original CDN-domain (e.g., `dCDN1.cdn.csp.com`), plus an NS record that maps `dCDN1.cdn.csp.com` to dCDN1's Request Router.
3. The end-user does a DNS lookup using the modified CDN-domain (i.e., `dCDN1.cdn.csp.com`). This causes dCDNn's request router returns an IP address of a suitable delivery node.
4. The end-user requests the content from dCDNn's delivery node. In the case of a cache hit, steps 5 ~ 7 do not happen, and the content data is directly returned by the delivery node to the end-user. In the case of a cache miss, the content needs to be acquired by dCDNn from either parent dCDN or uCDN (not the CSP). It also performs loop prevention process as described in the step 2 based on the provided CDN-Provider-ID.
5. Depending on the number of levels of redirection and availability of contents, the same process repeats until either content serving CDN provider is found or `MaxNumRedHops` expires.
6. Assuming that all intermediate dCDNs also miss cache, uCDN is selected as a content delivery CDN provider. Thus, the request router for uCDN selects a suitable delivery node to serve the inter-CDN acquisition request and returns IP address of the suitable uCDN delivery node.
7. uCDN serves content to dCDN1 and further down to end-user. Although not shown, it is at this point that uCDN processes the

rest of the URL: it extracts information identifying the origin server, validates that this server has been registered, and determines the content provider that owns the origin server.

3. Redirection Loop Prevention

According to the CDNi generic and request routing interface requirements, the CDNi solution shall support loop prevention of any CDN request routing redirection and subsequently allowing the request routing redirection. To meet such requirements, this section describes request routing redirection loop prevention mechanisms. Loop prevention mechanism should support both detection of the loop and post processing after loop detection. Also loop prevention should be applicable for the process of redirection CDN provider selection and inter CDN content acquisition.

This document specifies loop prevention mechanisms based on CDN-Provider-ID. Framework document [I-D.ietf-cdni-framework] recommends using distinguished acquisition domain for loop detection. It has some drawbacks such as overheads caused by length and processing time of domain name stacking in the case of cascading redirection. This document defines more general solution which can be applicable in both HTTP-based and DNS-based redirection. CDN-Provider-ID which is described in details in the section 2.1 is our proposed solution. Unlike distinguished domain name which is proposed in the framework, it is simple, unique, and efficient.

Post-processing after loop detection is also as important as its detection. For simplicity of the description, we provide a pseudo code of the post loop detection processing as below.

```
if (uCDN.ProviderList contains my.ProviderName or MaxNumRedHops <= 0)
{ //loop detected
  if (my.Avail == true) {
    request.DoService(my);
  } elseif(parent.Avail == true) {
    request.Redirect(parent);
  } elseif(uCDN.Avail == true) {
    request.Redirect(uCDN);
  } else {
    request.deny(); } }
```

If a loop is detected and myself can serve the request, the request is processed in my CDN. For some reason, if myself cannot process the request, it first checks availability of its parent and then uCDN. If none of them succeed, then request is denied.

4. Redirection Operational Considerations

For efficient request routing redirection, various operational considerations need to be addressed. In the framework document, for the redirection selection criteria, CDNi uses end-user's IP address prefix. However, in real CDN service environments, there are various other reasons such as service availability, QoS requirements, resource faults, etc. Both Routing Request Interface and redirection request should allow exchange of such information. The type and details of information that can be exchanged among CDN providers for the efficient redirection has to be considered together with footprint/capability advertisement. The details are for further study.

Performance feasibility of request routing redirection and loop prevention should be addressed. The requirements may vary depending on the CDN service types (e.g., CDN for small and/or large object). Also granularity of redirection within or between contents should be considered. It is for further study, too.

5 Security Considerations

6 IANA Considerations

7 References

7.1 Normative References

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[I-D.davie-cdni-framework] Peterson, L. and B. Davie, "Framework for CDN Interconnection", April 2012.

[I-D.ietf-cdni-requirements] Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", December 2011.

7.2 Informative References

[I-D.ietf-cdni-problem-statement] Niven-Jenkins, B., Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", May 2012.

[I-D.ietf-cdni-use-cases] Bertrand, G., Stephan, E., Burbridge, T., Eardley, P., Ma, K., and G. Watson, "Use Cases for Content Delivery Network Interconnection", June 2012.

Authors' Addresses

Taesang Choi
ETRI
161 Gajong-Dong
Yusong-Gu, Daejeon
Republic of Korea

Phone: +82-42-860-5628
Email: choits@etri.re.kr

Young-IL Seo
KT Network R&D Laboratory

463-1, Jeonmoin-dong,
Yuseong-gu, Daejeon
Republic of Korea

Phone: 82-10-3235-0135
Email: yohan.seo@kt.com

Dong-Ju Kim
KT Network R&D Laboratory
463-1, Jeonmoin-dong,
Yuseong-gu, Daejeon
Republic of Korea

Phone: 82-10-2686-9605
Email: dj.kim@kt.com

Jongmin Lee
SK Telecom
11, Euljiro-2ga
Jung-gu, Seoul
Republic of Korea

Phone: 82-10-9429-6260
Email: jminlee@sk.com

Ja-Ryeong Koo
LG U plus Corporation
Namdaemunro 5-ga
Jung-gu, Seoul
Republic of Korea

Phone: 82-10-8080-6115
Email: wjbkoo@lguplus.co.kr

John Dongho Shinn
Solbox Inc.
7F, Haesung Bldg. 747-2 Yeoksam-Dong
Kangnam-Gu, Seoul
Republic of Korea

Phone: +82-10-3005-4785
Email: eastsky@solbox.com

INTERNET-DRAFT
Intended Status: Standards Track
Expires: April 24, 2013

TS Choi
ETRI
YI Seo
DJ Kim
KT
JM Lee
SKT
JR Koo
LGU+
JDH Shinn
Solbox Inc.
KS Park
KAIST
October 21, 2012

CDNi Request Routing Redirection with Loop Prevention
draft-choi-cdni-req-routing-redir-loop-prevention-01

Abstract

This document describes request routing redirection procedures, loop prevention mechanisms, and other operational considerations which are associated with redirection.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 29, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1	Introduction	3
1.1	Terminology	3
2.	Redirection Procedures	3
2.1.	Preliminaries	3
2.2.	Iterative Redirection Procedures	4
2.2.1.	HTTP-based Redirection	4
2.2.2.	DNS-based Redirection	8
2.3.	Recursive Redirection Procedures	10
2.3.1.	HTTP-based Redirection	10
2.3.2.	DNS-based Redirection	14
3.	Redirection Loop Prevention	17
4.	Redirection Operational Considerations	18
5	Security Considerations	19
6	IANA Considerations	19
7	References	19
7.1	Normative References	19
7.2	Informative References	19
	Authors' Addresses	19

1 Introduction

According to the CDNi generic and request routing interface requirements[I-D.ietf-cdni-requirements], the CDNi solution shall support iterative and recursive CDNi request routing, efficient request routing for small and large objects, arbitrary number of levels of cascaded CDN redirection, looping prevention of any CDN request routing redirection, and subsequently allowing the request routing redirection. To meet such requirements, this document describes request routing redirection procedures, loop prevention mechanisms, and other operational considerations that are associated with redirection.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Redirection Procedures

2.1. Preliminaries

To meet the requirements of request routing redirection, we define the term "CDN-Provider-ID". It uniquely identifies each CDN provider during the course of request routing redirection. It consists of "CDN provider name" and "MaxNumRedHops". A pair of an AS number and an additional qualifier is used for CDN provider name. Since more than one CDN providers can belong to the same AS, an additional qualifier is used to guarantee the uniqueness. MaxNumRedHops represents a maximum allowed redirections. The value is decreased once every redirection occurs until it reaches 0. To avoid its usage abuse (e.g., end user or CDN operator can set huge number like 100 or above), a reasonable upper bound has to be agreed among CDN providers. Security aspect of it is for further study.

A few examples of the CDN provider names are 100:0 and 200:1. The former means that a CDN provider belong to AS 100 and it is the only CDN provider within that AS. The latter represents the first CDN provider in the AS 200. There are other CDN providers in the same AS.

One example of CDN-Provider-ID is "CDN-Provider-Name=100:0 & MaxNumRedHops=10", which means that a CDN provider that belong to AS number 100 and it is the only CDN provider and a maximum allowed redirection is 10. An example how a list of CDN-Provider-IDs can be carried in the URI query string when a certain cascaded request

routing redirection occurs is the following. We assume that redirection is cascaded three times: uCDN -> dCDN1 -> dCDN2. dCDN1, then, carries the following URL, "http://cdn.csp.com?uCDN-Provider-ID=100:0&dCDN1-Provider-ID=200:1&MaxNumRedHops=9". Note that MaxNumRedHops carries the latest number instead of adding in every CDN-Provider-ID to save the space in URI query string.

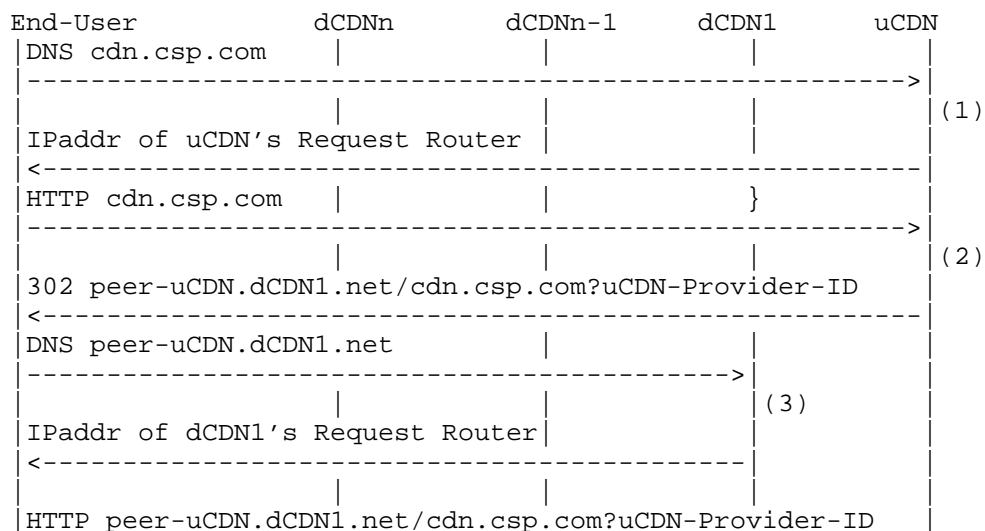
It is applicable for both HTTP-based and DNS redirections. For HTTP-based redirection, we define a HTTP request routing redirection header "CDN-Provider-ID". For each step of redirection, it is attached to the beginning of the provider domain URL. For example, uCDN initiates a redirection with its URL, http://100:0:10.cdn.csp.com. dCDN further attaches its own CDN-Provider-ID in the front when another level of redirection is required. For DNS-based redirection, the CDN-Provider-ID can be attached in the DNS CNAME.

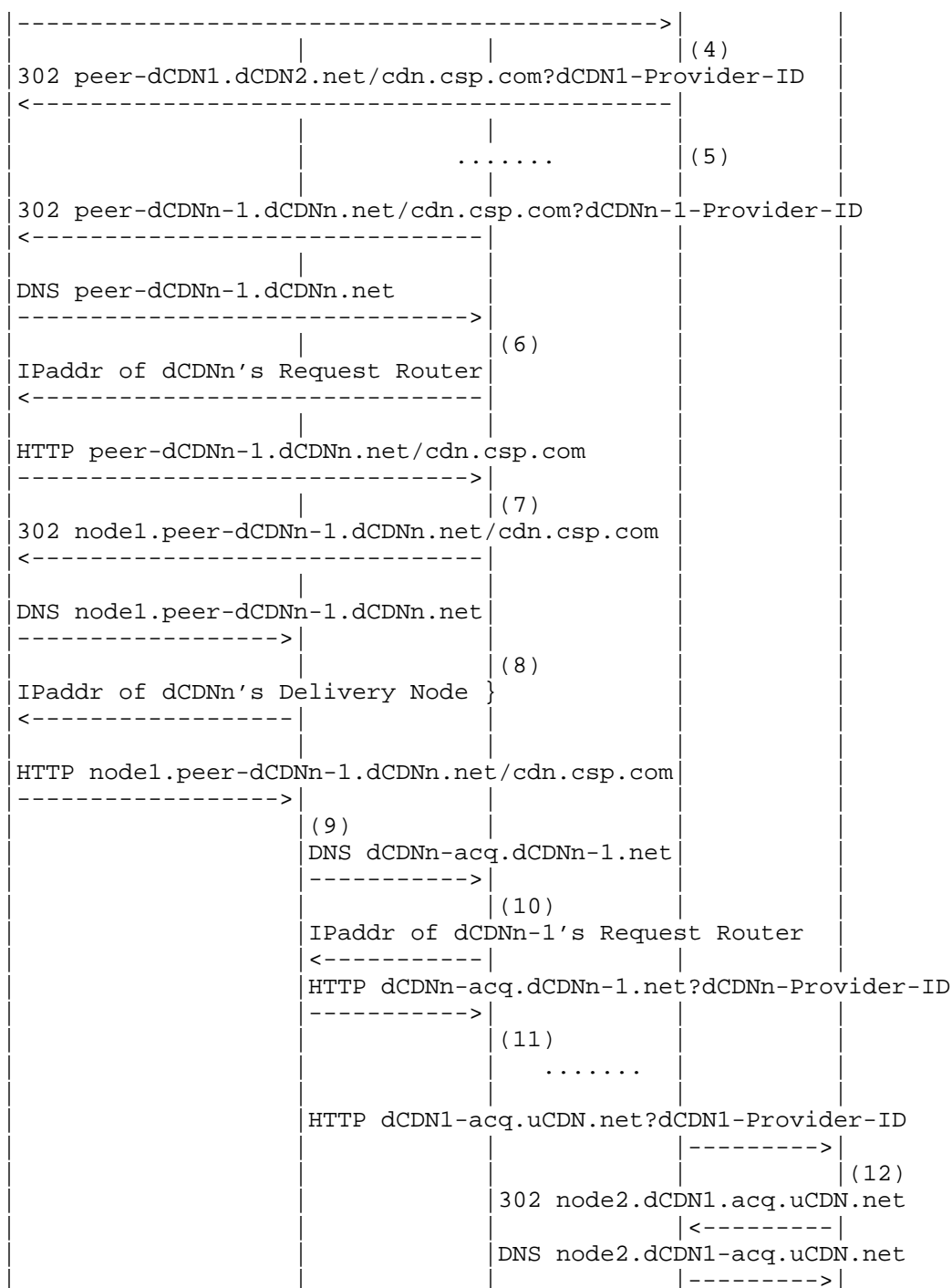
Since there is a CDNi requirement to support of arbitrary topology of interconnected CDNs, this document assumes that the redirection procedures and loop prevention mechanisms must also support arbitrary topology.

2.2. Iterative Redirection Procedures

2.2.1. HTTP-based Redirection

In this section, we describe an iterative procedure of HTTP-based request routing redirection with loop prevention.





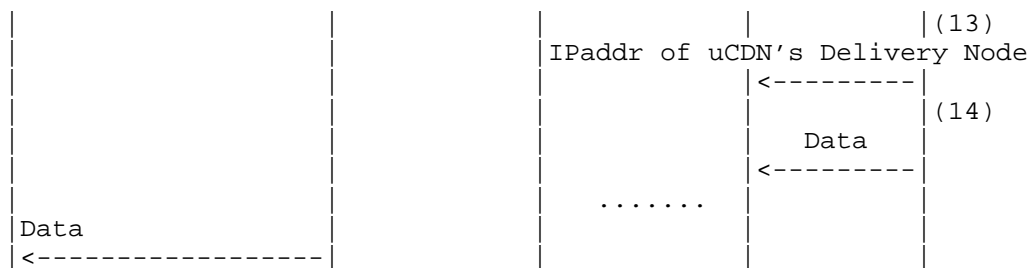


Figure 1: HTTP-based request routing redirection iterative procedure

The steps illustrated in the figure are as follows:

1. A DNS resolver for uCDN provider processes the DNS request for its customer based on CDN-domain `cdn.csp.com`. It returns the IP address of a request router in uCDN provider.
2. A Request Router for uCDN provider processes the HTTP request and recognizes that the end-user is best served by dCDN1. So it returns a 302 redirect message for a new URL constructed by "stacking" dCDN1's distinguished CDN-domain (`peer-uCDN.dCDN1.net`) on the front of the original URL. It also adds uCDN's CDN-Provider-ID in the URI query string of the HTTP request message. (e.g., `uCDN-Provider-ID=100:0 & MaxNumRedHops=10`). This information is not processed by the customer but conveyed in the HTTP message without any modification of the step 4. The details on how it is used for loop prevention is described in the step 4.
3. The end-user does a DNS lookup using dCDN1's distinguished CDN-domain (`peer-uCDN.dCDN1.net`). dCDN1's DNS resolver returns the IP address of a request router for dCDN1.
4. The request router for dCDN1 processes the HTTP request. There are two options: redirect further to another dCDN (i.e., cascading the request) or process it by itself. In either cases, it performs loop prevention step first. It checks a list of CDN-provider-IDs in the URI query string: it contains a list of CDN providers which requested redirections so far. If either it contains own CDN provider name or `MaxNumRedHops` becomes 0, it means that the redirection loop has occurred or the number of redirection hops has reached the maximum. Once loop is detected, details on the next steps is described in the section 3. If it is loop free, it either redirects further or processes based on the local policy. For the former, it selects another dCDN provider and sends an HTTP redirect message with its own CDN-Provider-ID included in its URI query string (e.g.,

uCDN-Provider-ID=100:0 & dCDN1-Provider-ID=200:1 & MaxNumRedHops=9) attached. For the latter, it selects a suitable delivery node to serve the end-user request, and returns a 302 redirect message for a new URL constructed by replacing the hostname by a subdomain of the dCDN1's distinguished CDN-domain that points to the selected delivery node. Then it goes to the step 6.

5. If further redirection is decided, it repeats steps 2 - 4 until it either selects dCDN provider to serve the request or MaxNumRedHops expires. If the former occurs, it resumes the step 6. If the latter occurs, it follows the processes described in the section 3.
6. Assuming that dCDNn is selected as a serving dCDN provider, the end-user does a DNS lookup using dCDNn's distinguished CDN-domain (peer-dCDNn-1.dCDNn.net). dCDNn-1's DNS resolver returns the IP address of a request router for dCDNn.
7. The request router for dCDN1 processes the HTTP request and selects a suitable delivery node to serve the end-user request, and returns a 302 redirect message for a new URL constructed by replacing the hostname by a subdomain of the dCDNn's distinguished CDN-domain that points to the selected delivery node.
8. The end-user does a DNS lookup using dCDNn's delivery node subdomain (node1.peer-dCDNn-1.dCDNn.net). dCDNn's DNS resolver returns the IP address of the delivery node.
9. The end-user requests the content from dCDNn's delivery node. In the case of a cache hit, steps 10 ~ 14 below do not happen, and the content data is directly returned by the delivery node to the end-user. In the case of a cache miss, the content needs to be acquired by dCDN from either parent dCDN or uCDN (not the CSP). The distinguished CDN-domain peer-dCDNn-1.dCDNn.net indicates that this content is to be acquired from dCDNn-1; stripping the CDN-domain reveals the original CDN-domain cdn.csp.com and dCDNn may verify that this CDN-domain belongs to a known peer (so as to avoid being tricked into serving as an open proxy). It then does a DNS request for an inter-CDN acquisition CDN-domain as agreed above (in this case, dCDNn-acq.dCDNn-1.net). This process repeats recursively until it finds a CDN provider that can serve the requested content.
10. dCDNn-1's DNS resolver processes the DNS request and returns the IP address of a request router in dCDNn-1.

11. The request router for dCDNn-1 processes the HTTP request from dCDNn's delivery node. dCDNn-1 request router recognizes that the request is from a peer CDN rather than an end-user because of the dedicated inter-CDN acquisition domain (dCDNn-acq.dCDNn-1.net). It also performs loop prevention process as described in step 4 based on the provided CDN-Provider-ID (e.g., uCDN-Provider-ID=100:0 & dCDN1-Provider-ID=200:1 & ... & dCDNn-Provider-ID=1000:0 & MaxNumRedHops=1). Depending on the number of levels of redirection and availability of contents, the same process repeats until either content serving CDN provider is found or MaxNumRedHops expires.
12. Assuming that all intermediate dCDNs also have a cache miss, The request router for uCDN selects a suitable delivery node to serve the inter-CDN acquisition request and returns a 302 redirect message for a new URL constructed by replacing the hostname by a subdomain of the uCDN's distinguished inter-CDN acquisition domain that points to the selected delivery node.
13. uCDN DNS resolver processes the DNS request and returns the IP address of the delivery node in uCDN.
14. uCDN serves content for the requested CDN-domain to dCDN and finally to end-user. Although not shown, it is at this point that uCDN processes the rest of the URL: it extracts information identifying the origin server, validates that this server has been registered, and determines the content provider that owns the origin server. It may also perform its own content acquisition steps if needed before returning the content to dCDN.

2.2.2. DNS-based Redirection

In this section, we describe an iterative procedure of DNS-based request routing redirection with loop prevention.

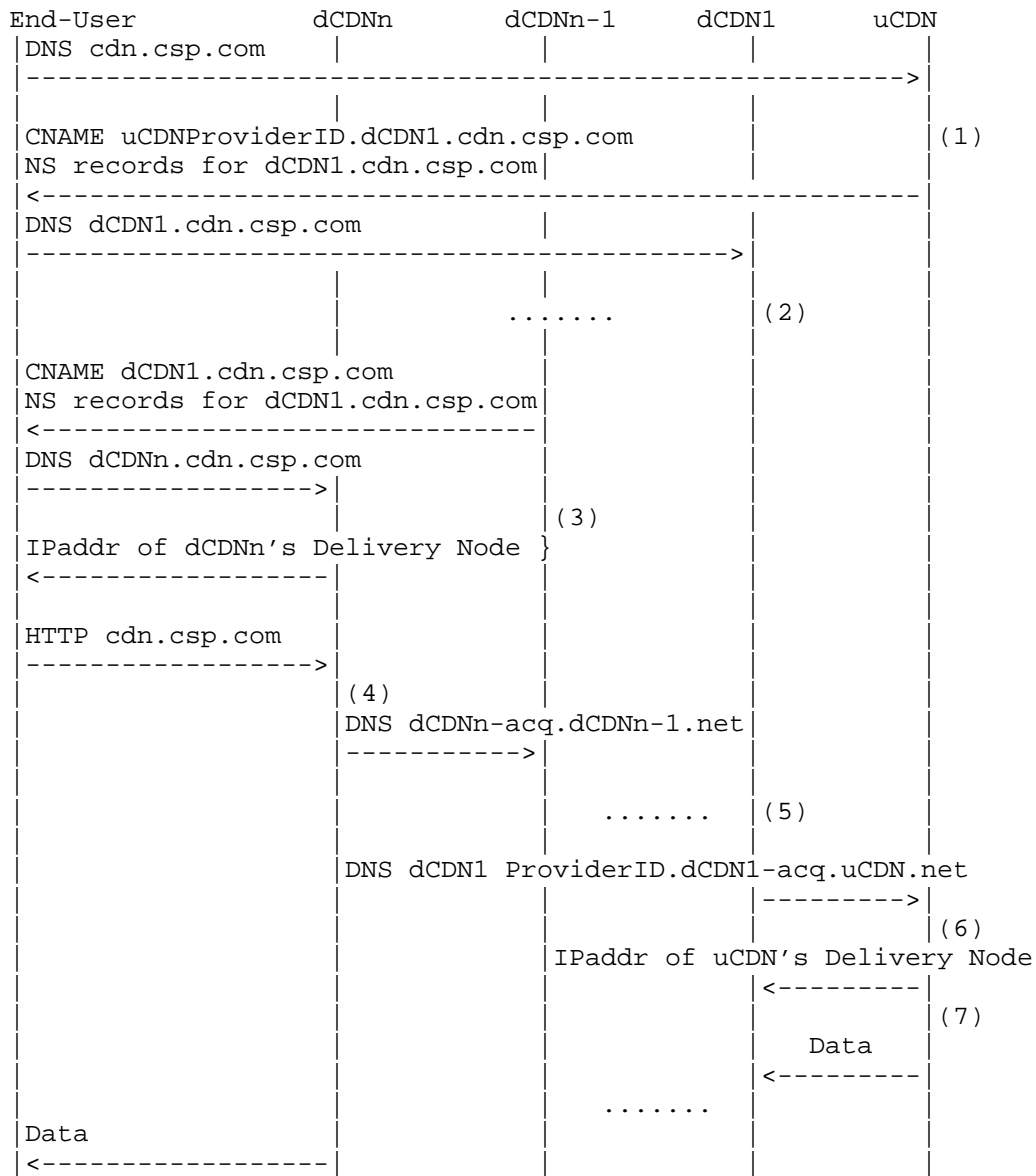


Figure 2: DNS-based request routing redirection iterative procedure

The steps illustrated in the figure are as follows:

1. Request Router for uCDN provider processes the DNS request for CDN- domain `cdn.csp.com` and recognizes that the end-user is best

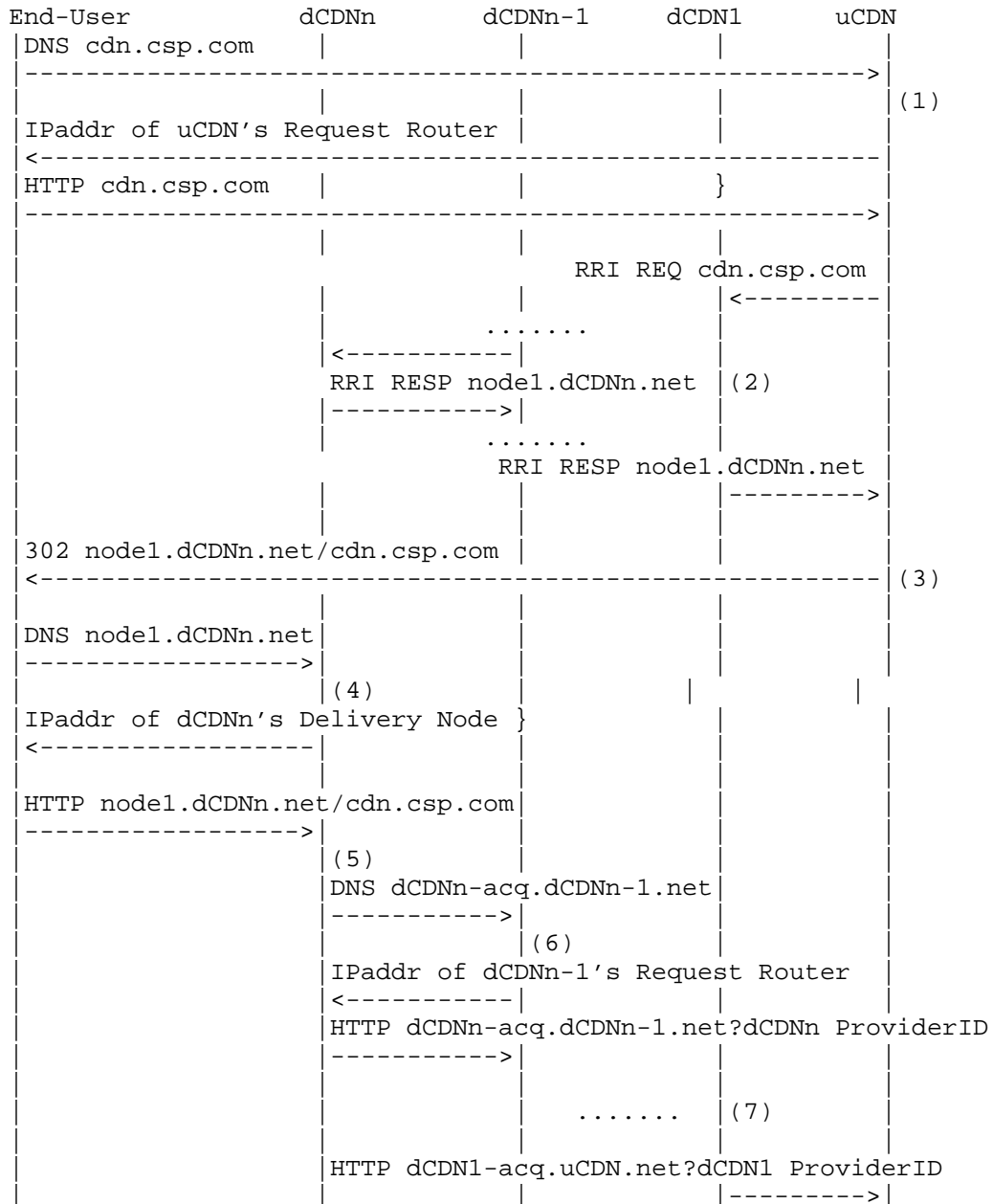
served by another CDN. (This may depend on the IP address of the user's local DNS resolver, or other information discussed below.) The Request Router returns a DNS CNAME response by "stacking" the distinguished identifier for dCDN1 and uCDN's CDN-Provider-ID (e.g., 100:0.10) onto the original CDN-domain (e.g., dCDN1.cdn.csp.com), plus an NS record that maps dCDN1.cdn.csp.com to dCDN1's Request Router.

2. The end-user does a DNS lookup using the modified CDN-domain (i.e., dCDN1.cdn.csp.com). dCDN1 Request Router processes the request and decides to serve the request or redirect further to another CDN provider. It also checks redirection loop. This process iterates until either serving dCDN is selected or MaxNumRedHops expires. In this case, dCDNn is selected as a serving dCDN. If the former occurs, it proceeds to step 3. If the latter occurs, it follows the processes described in the section 3.
3. The end-user does a DNS lookup using the modified CDN-domain (i.e., dCDN1.cdn.csp.com). This causes dCDNn's request router returns an IP address of a suitable delivery node.
4. The end-user requests the content from dCDNn's delivery node. In the case of a cache hit, steps 5 ~ 7 do not happen, and the content data is directly returned by the delivery node to the end-user. In the case of a cache miss, the content needs to be acquired by dCDNn from either parent dCDN or uCDN (not the CSP). It also performs loop prevention process as described in the step 2 based on the provided CDN-Provider-ID (e.g., 100:0.200:1.....900:0.1)
5. Depending on the number of levels of redirection and availability of contents, the same process repeats until either content serving CDN provider is found or MaxNumRedHops expires.
6. Assuming that all intermediate dCDNs also miss cache, uCDN is selected as a content delivery CDN provider. Thus, the request router for uCDN selects a suitable delivery node to serve the inter-CDN acquisition request and returns IP address of the suitable uCDN delivery node.
7. uCDN serves content to dCDN1 and further down to end-user. Although not shown, it is at this point that uCDN processes the rest of the URL: it extracts information identifying the origin server, validates that this server has been registered, and determines the content provider that owns the origin server.

2.3. Recursive Redirection Procedures

2.3.1. HTTP-based Redirection

In this section, we describe an recursive procedure of HTTP-based request routing redirection with loop prevention.



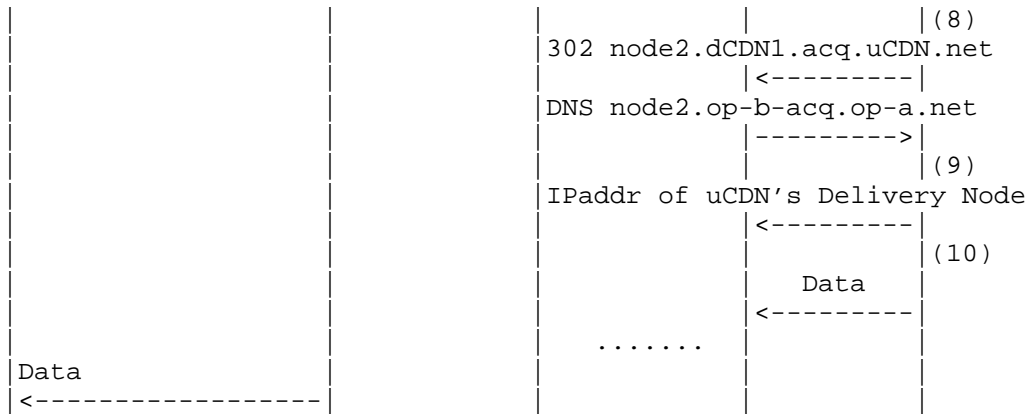


Figure 3: HTTP-based request routing redirection recursive procedure

The steps illustrated in the figure are as follows:

1. A DNS resolver for uCDN provider processes the DNS request for its customer based on CDN-domain `cdn.csp.com`. It returns the IP address of a request router in uCDN provider.
2. A Request Router for uCDN provider processes the HTTP request and recognizes that the end-user is best served by dCDN1. So it queries the CDNI Request Routing interface of dCDN1 providing a set of information about the request including the URL requested. It also provides uCDN's CDN-Provider-ID (e.g., uCDN-Provider-ID=100:0 & MaxNumRedHops=10) for loop prevention process. It contains a list of CDN providers that have requested redirections so far. If either it contains its own CDN provider name or MaxNumRedHops becomes 0, it means that the redirection loop has occurred or it has reached the maximum number of allowed number of redirection hops. Once loop is detected, details on the next steps are described in the section 3. If it is loop free, dCDN1 then either replies with the DNS name of a delivery node or redirect to another dCDN. Such cascading redirection can continue until a serving dCDN is decided. The RRI RESP can be sent in the reverse order of cascaded redirection or directly to the redirection origin CDN provider if contact information is known. The contact information can be embedded in the RRI REQ message or pre-configured during bootstrapping process. The default behavior is recursive RESP.
3. uCDN returns a 302 redirect message for a new URL obtained from

the Request Routing Interface.

4. The end-user does a DNS lookup using the host name of the URL just provided (node1.dCDNn.net). dCDNn's DNS resolver returns the IP address of the corresponding delivery node. Note that, since the name of the delivery node was already obtained from dCDNn using the CDNI Request Routing Interface, there should not be any further redirection here (in contrast to the iterative method described above.)
4. The request router for dCDN1 processes the HTTP request. There are two options: redirect further to another dCDN (i.e., cascading the request) or process it by itself. In either cases, it performs loop prevention step first. If it is loop free, it either redirect further or processes based on the local policy. For the former, it selects another dCDN provider and sends HTTP redirect message with its own CDN-Provider-ID (e.g., uCDN-Provider-ID=100:0 & dCDN1-Provider-ID=200:1 & MaxNumRedHops=9) attached. For the latter, it selects a suitable delivery node to serve the end-user request, and returns a 302 redirect message for a new URL constructed by replacing the hostname by a subdomain of the dCDN1's distinguished CDN-domain that points to the selected delivery node. Then it goes to step 6.
5. The end-user requests the content from dCDNn's delivery node. In the case of a cache hit, steps 6 ~ 10 below do not happen, and the content data is directly returned by the delivery node to the end-user. In the case of a cache miss, the content needs to be acquired by dCDN from either parent dCDN or uCDN (not the CSP). The distinguished CDN-domain dCDNn.net indicates that this content is to be acquired from dCDNn-1; stripping the CDN-domain reveals the original CDN-domain cdn.csp.com and dCDNn may verify that this CDN-domain belongs to a known peer (so as to avoid being tricked into serving as an open proxy). It then does a DNS request for an inter-CDN acquisition CDN-domain as agreed above (in this case, dCDNn-acq.dCDNn-1.net). This process repeats recursively until it finds CDN provider that can serve the requested content.
6. dCDNn-1's DNS resolver processes the DNS request and returns the IP address of a request router in dCDNn-1.
7. The request router for dCDNn-1 processes the HTTP request from dCDNn's delivery node. dCDNn-1 request router recognizes that the request is from a peer CDN rather than an end-user because of the dedicated inter-CDN acquisition domain

(dCDNn-acq.dCDNn-1.net). It also performs loop prevention process as described in the step 2 based on the provided CDN-Provider-ID (e.g., uCDN-Provider-ID=100:0 & dCDN1-Provider-ID=200:1 & ... & dCDNn-1-Provider-ID=900:0 & MaxNumRedHops=1). Depending on the number of levels of redirection and availability of contents, the same process repeats until either content serving CDN provider is found or MaxNumRedHops expires.

8. Assuming that all intermediate dCDNs also have a cache miss, The request router for uCDN selects a suitable delivery node to serve the inter-CDN acquisition request and returns a 302 redirect message for a new URL constructed by replacing the hostname by a subdomain of the uCDN's distinguished inter-CDN acquisition domain that points to the selected delivery node.
9. uCDN DNS resolver processes the DNS request and returns the IP address of the delivery node in uCDN.
10. uCDN serves content for the requested CDN-domain to dCDN and finally to end-user. Although not shown, it is at this point that uCDN processes the rest of the URL: it extracts information identifying the origin server, validates that this server has been registered, and determines the content provider that owns the origin server. It may also perform its own content acquisition steps if needed before returning the content to dCDN.

2.3.2. DNS-based Redirection

In this section, we describe an recursive procedure of DNS-based request routing redirection with loop prevention.

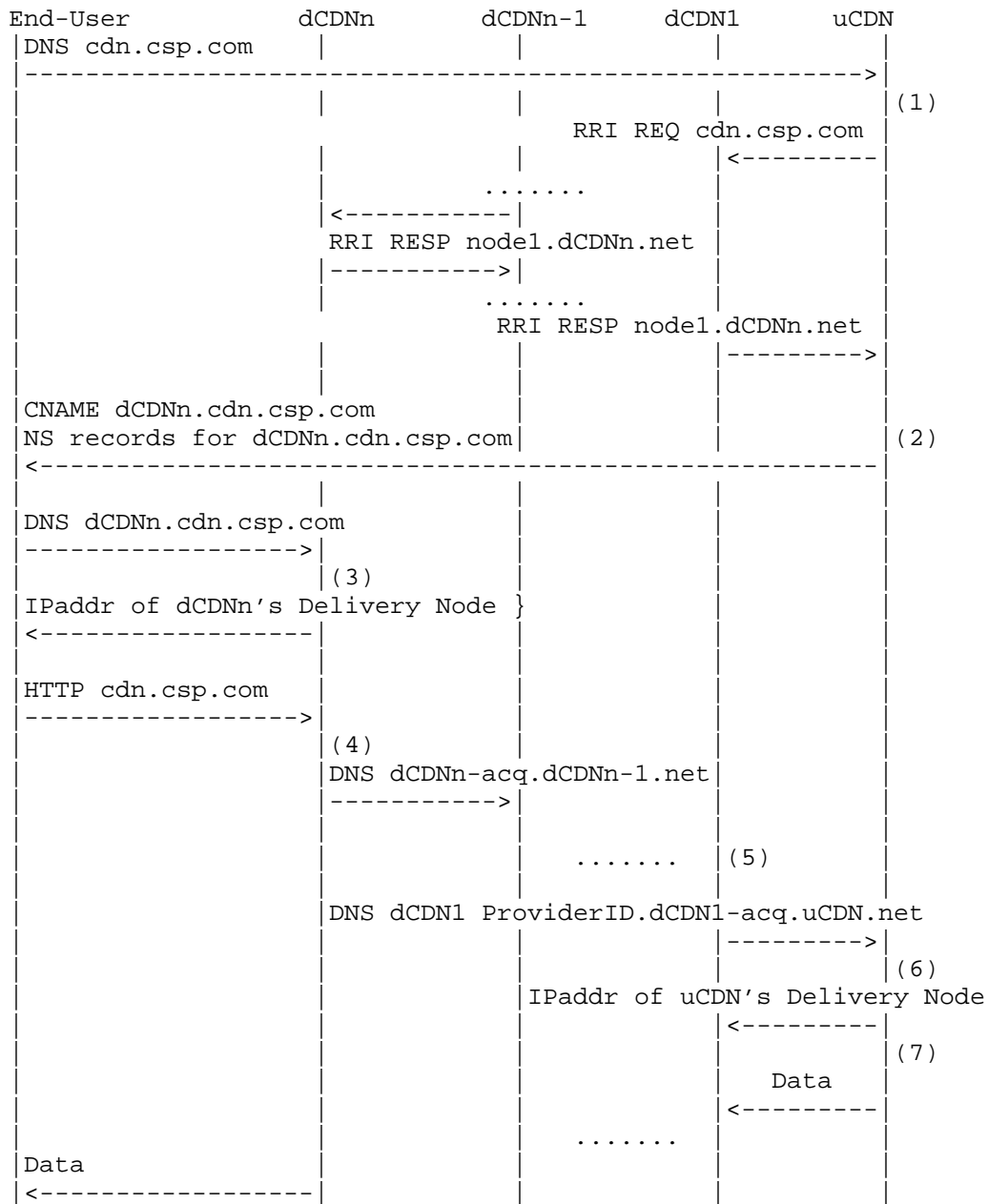


Figure 4: DNS-based request routing redirection recursive procedure

The steps illustrated in the figure are as follows:

1. Request Router for uCDN provider processes the DNS request for CDN-domain `cdn.csp.com` and recognizes that the end-user is best served by dCDN1. So it queries the CDNI Request Routing Interface of dCDN1 providing a set of information about the request including the URL requested. It also provides uCDN's CDN-Provider-ID (e.g., `uCDN-Provider-ID=100:0` & `MaxNumRedHops=10`) for loop prevention process. It checks CDN-Provider-ID. If either it contains own CDN provider name or `MaxNumRedHops` becomes 0, it means that the redirection loop is occurred. Once loop is detected, details on the next steps are described in the section 3. If it is loop-free, dCDN1 then either replies with the DNS name of a delivery node or redirect to another dCDN. Such cascaded redirection can continue until a serving dCDN is decided. The RRI RESP can be sent in the reverse order of cascaded redirection or directly to the redirection origin CDN provider if contact information is known. The contact information can be embedded in the RRI REQ message or pre-configured during bootstrapping process. The default behavior is recursive RESP. In this case, dCDNn is selected as a serving dCDN.
2. The Request Router returns a DNS CNAME response by "stacking" the distinguished identifier for dCDNn and uCDN's CDN-Provider-ID onto the original CDN-domain (e.g., `dCDN1.cdn.csp.com`), plus an NS record that maps `dCDN1.cdn.csp.com` to dCDN1's Request Router.
3. The end-user does a DNS lookup using the modified CDN-domain (i.e., `dCDN1.cdn.csp.com`). This causes dCDNn's request router returns an IP address of a suitable delivery node.
4. The end-user requests the content from dCDNn's delivery node. In the case of a cache hit, steps 5 ~ 7 do not happen, and the content data is directly returned by the delivery node to the end-user. In the case of a cache miss, the content needs to be acquired by dCDNn from either parent dCDN or uCDN (not the CSP). It also performs loop prevention process as described in the step 2 based on the provided CDN-Provider-ID (e.g., `uCDN-Provider-ID=100:0` & `dCDN1-Provider-ID=200:1` & ... & `dCDNn-1-Provider-ID=900:1` & `MaxNumRedHops=1`).
5. Depending on the number of levels of redirection and availability of contents, the same process repeats until either content serving CDN provider is found or `MaxNumRedHops` expires.
6. Assuming that all intermediate dCDNs also miss cache, uCDN is selected as a content delivery CDN provider. Thus, the request router for uCDN selects a suitable delivery node to serve the inter-CDN acquisition request and returns IP address of the suitable uCDN delivery node.

7. uCDN serves content to dCDN1 and further down to end-user. Although not shown, it is at this point that uCDN processes the rest of the URL: it extracts information identifying the origin server, validates that this server has been registered, and determines the content provider that owns the origin server.

3. Redirection Loop Prevention

According to the CDNi generic and request routing interface requirements, the CDNi solution shall support loop prevention of any CDN request routing redirection and subsequently allowing the request routing redirection. To meet such requirements, this section describes request routing redirection loop prevention mechanisms. Loop prevention mechanism should support both detection of the loop and post processing after loop detection. Also loop prevention should be applicable for the process of redirection CDN provider selection and inter CDN content acquisition.

This document specifies loop prevention mechanisms based on CDN-Provider-ID. Framework document [I-D.ietf-cdni-framework] recommends using distinguished acquisition domain for loop detection. It has some drawbacks such as overheads caused by length and processing time of domain name stacking in the case of cascading redirection. This document defines more general solution which can be applicable in both HTTP-based and DNS-based redirections. CDN-Provider-ID which is described in details in the section 2.1 is our proposed solution. Unlike distinguished domain name which is proposed in the framework, it is simple, unique, and efficient.

Post-processing after loop detection is also as important as its detection. The most strict option is to deny the service when the loop is detected. This option should be the last resort when other alternatives are not available. The simplest choice is the CDN provider which detected the loop provide the service by itself although the service quality may not be optimal. If it is not possible, it requests its parent. If the parent can provide the service, it does so. If not, it further checks with other dCDN providers at the peer level or downstream until it finds available one. If it all fails its attempts at the its parent, peer, and children, it checks with uCDN. If note is available, it finally rejects the service.

4. Redirection Operational Considerations

For efficient request routing redirection, various operational considerations need to be addressed. In the framework document, for the redirection selection criteria, CDNi uses end-user's IP address prefix. However, in real CDN service environments, there are various

other reasons such as service availability, QoS requirements, resource faults, etc. Both Routing Request Interface and redirection request should allow exchange of such information. The type and details of information that can be exchanged among CDN providers for the efficient redirection has to be considered together with footprint/capability advertisement. The details are for further study.

Performance feasibility of request routing redirection and loop prevention should be addressed. The requirements may vary depending on the CDN service types (e.g., CDN for small and/or large object). Also granularity of redirection within or between contents should be considered. It is for further study, too.

5 Security Considerations

6 IANA Considerations

7 References

7.1 Normative References

- [KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [I-D.davie-cdni-framework] Peterson, L. and B. Davie, "Framework for CDN Interconnection", April 2012.
- [I-D.ietf-cdni-requirements] Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", December 2011.

7.2 Informative References

- [I-D.ietf-cdni-problem-statement] Niven-Jenkins, B., Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", May 2012.
- [I-D.ietf-cdni-use-cases] Bertrand, G., Stephan, E., Burbridge, T., Eardley, P., Ma, K., and G. Watson, "Use Cases for Content Delivery Network Interconnection", June 2012.

Authors' Addresses

Taesang Choi
ETRI
161 Gajong-Dong
Yusong-Gu, Daejeon
Republic of Korea

Phone: +82-42-860-5628
Email: choits@etri.re.kr

Young-IL Seo
KT Network R&D Laboratory

463-1, Jeonmoin-dong,
Yuseong-gu, Daejeon
Republic of Korea

Phone: 82-10-3235-0135
Email: yohan.seo@kt.com

Dong-Ju Kim
KT Network R&D Laboratory
463-1, Jeonmoin-dong,
Yuseong-gu, Daejeon
Republic of Korea

Phone: 82-10-2686-9605
Email: dj.kim@kt.com

Jongmin Lee
SK Telecom
11, Euljiro-2ga
Jung-gu, Seoul
Republic of Korea

Phone: 82-10-9429-6260
Email: jminlee@sk.com

Ja-Ryeong Koo
LG U plus Corporation
Namdaemunro 5-ga
Jung-gu, Seoul
Republic of Korea

Phone: 82-10-8080-6115
Email: wjbkoo@lguplus.co.kr

John Dongho Shinn
Solbox Inc.
7F, Haesung Bldg. 747-2 Yeoksam-Dong
Kangnam-Gu, Seoul
Republic of Korea

Phone: +82-10-3005-4785
Email: eastsky@solbox.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 10, 2013

B. Niven-Jenkins
R. Murray
G. Watson
Velocix (Alcatel-Lucent)
M. Caulfield
K. Leung
Cisco Systems
July 9, 2012

CDN Interconnect Metadata
draft-cjlmw-cdni-metadata-00

Abstract

The CDNI Metadata Interface enables interconnected CDNs to exchange content distribution metadata in order to enable content acquisition and delivery. The CDNI metadata associated with a piece of content provides a downstream CDN with sufficient information for the downstream CDN to service content requests on behalf of an upstream CDN. This document describes both the core set of CDNI metadata and the protocol for exchanging that metadata.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 10, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
1.1. Terminology	4
2. Design Principles	5
3. CDNI Metadata Data Model	5
3.1. HostIndex, HostMetadata & PathMetadata objects	6
3.2. Remaining CDNI Metadata objects	9
3.3. Metadata Inheritance	12
4. Encoding-Independent CDNI Metadata Object Descriptions	12
4.1. CDNI Metadata Data Object Descriptions	12
4.1.1. HostIndex	13
4.1.2. HostMatch	13
4.1.3. HostMetadata	13
4.1.4. Acquisition	14
4.1.5. Delivery	14
4.1.6. PathMatch	15
4.1.7. PathMetadata	15
4.1.8. ACL	15
4.1.9. ACLRule	16
4.1.10. TimeWindow	16
4.1.11. Location	17
4.1.12. Source	17
4.1.13. Auth	17
4.1.14. Link	18
4.2. CDNI Metadata Simple Data Type Descriptions	19
4.2.1. Protocol	19
4.2.2. Endpoint	19
4.2.3. IPRange	19
4.2.4. Pattern	20
4.2.5. PatternFlags	20
4.2.6. URI	20
4.2.7. Time	20
5. CDNI Metadata interface	21

5.1.	Transport	21
5.2.	Retrieval of CDNI Metadata resources	22
5.3.	Bootstrapping	23
5.4.	Encoding	23
5.4.1.	MIME Media Types	23
5.4.2.	JSON Encoding of Objects	24
5.4.2.1.	JSON Example	25
5.4.3.	XML Encoding of Objects	27
5.4.3.1.	XML Example	28
5.5.	Extensibility	30
6.	IANA Considerations	31
7.	Security Considerations	31
8.	Acknowledgements	31
9.	References	31
9.1.	Normative References	31
9.2.	Informative References	31
	Appendix A. Relationship to the CDNI Requirements	32
	Authors' Addresses	33

1. Introduction

CDNI enables a downstream CDN to service content requests on behalf of an upstream CDN. In the simplest use case, a content request received by the downstream CDN provides sufficient information for sending a response. More complex use cases require additional context, i.e. metadata. The CDNI metadata associated with a piece of content (or with a set of contents) provides a downstream CDN with sufficient information for servicing content requests on behalf of an upstream CDN in accordance with the policies defined by the upstream CDN.

The CDNI Metadata Interface is introduced by [I-D.ietf-cdni-problem-statement] along with three other interfaces that may be used to compose a CDNI solution (Control, Request Routing and Logging). [I-D.davie-cdni-framework] expands on the information provided in [I-D.ietf-cdni-problem-statement] and describes each interface, and the relationships between them, in more detail. The requirements for the CDNI metadata interface are specified in [I-D.ietf-cdni-requirements]

This document focuses on the CDNI Metadata interface which enables a downstream CDN to obtain CDNI Metadata from an upstream CDN so that the downstream CDN can properly process and respond to:

- o Redirection Requests received over the CDNI Request Routing protocol.
- o Content Requests received directly from User Agents.

Specifically this document proposes:

- o A data structure for mapping content requests to CDNI Metadata properties (Section 3).
- o An initial set of CDNI Metadata properties (Section 4.1 through Section 4.2).
- o A RESTful web service for the transfer of CDNI Metadata (Section 5).

1.1. Terminology

This document reuses the terminology defined in [I-D.ietf-cdni-problem-statement].

Additionally, the following terms are used throughout this document and are defined as follows:

- o Object - a collection of properties

- o Property - a key / value pair where the key is a property name and the value is the property value (possibly an object)

2. Design Principles

The proposed CDNI Metadata Interface aims to achieve the following design principles:

1. Cacheability of CDNI metadata objects
2. Deterministic mapping from content requests to CDNI metadata properties
3. Support for DNS redirection as well as application-specific redirection (for example HTTP redirection)
4. Minimal duplication of CDNI metadata
5. Leverage existing protocols

Cacheability improves the latency of acquiring metadata and therefore improves the latency of serving content requests. The CDNI Metadata Interface uses HTTP to achieve cacheability.

Deterministic mappings from content requests to metadata properties eliminates ambiguity and ensures that the same policies are applied consistently by all downstream CDNs.

Support for both HTTP and DNS redirection ensures that the CDNI Metadata Interface can be used for HTTP and DNS redirection and also meets the same design principles for both HTTP and DNS based redirection schemes.

Minimal duplication of CDNI metadata provides space efficiency on storage in the CDNs, on caches in the network, and across the network between CDNs.

Leveraging existing protocols avoids reinventing common mechanisms such as data structure encoding (e.g. XML, JSON) and data transport (e.g. HTTP).

3. CDNI Metadata Data Model

The CDNI Metadata Model describes a data structure for mapping content requests to metadata properties. Metadata properties describe how to acquire, authorize, and deliver content from a downstream CDN. The data model relies on the assumption that these metadata properties may be aggregated based on the authoritative hostname of the content and subsequently on the resource path of the content. The data model associates a set of CDNI Metadata properties

with a Hostname to form a default set of metadata properties for content delivered for that Hostname. That default set of metadata properties can be overridden by properties that apply to specific paths within a URI.

Different Hostnames and URI paths will contain different sets of CDNI Metadata properties in order to describe the required behaviour when a dCDN surrogate is processing User Agent requests for content on that Hostname or URI path. As a result of this structure, significant commonality may exist between the CDNI Metadata properties specified for different Hostnames, different URI paths within a Hostname and different URI paths on different Hostnames. For example the definition of which User Agent IP addresses should be treated as being grouped together into a single network or geographic location is likely to be common for a number of different Hostnames. Another example is that although a uCDN is likely to have several different policies configured to express geo-blocking rules, it is likely that a single geo-blocking policy would be applied to multiple Hostnames delivered through the CDN.

In order to enable the CDNI Metadata for a given Hostname or URI Path to be decomposed into sets of CDNI Metadata properties that can be reused by multiple Hostnames and URI Paths the CDNI Metadata interface specified in this document splits the CDNI Metadata into a number of objects. Efficiency is improved by enabling a single CDNI Metadata object (that is shared across Hostname and/or URI paths) to be retrieved by a dCDN once, even if it is referenced by the CDNI Metadata of multiple Hostnames.

Section 3.1 introduces a high level description of the HostIndex, HostMetadata and PathMetadata objects and describes the relationships between those objects.

Section 3.2 introduces a high level description of the remaining CDNI Metadata objects and describes the relationships between those objects as well as the relationships of those objects to HostMetadata and PathMetadata objects.

Section 4.1 describes the specific properties of each object in more detail.

3.1. HostIndex, HostMetadata & PathMetadata objects

A HostIndex object contains a list of Hostnames that may be delegated to the downstream CDN. The HostIndex is the starting point for accessing the uCDN's CDNI Metadata data store. It enables surrogates in the dCDN to deterministically discover, on receipt of a User Agent request for content, which other CDNI Metadata objects it requires in

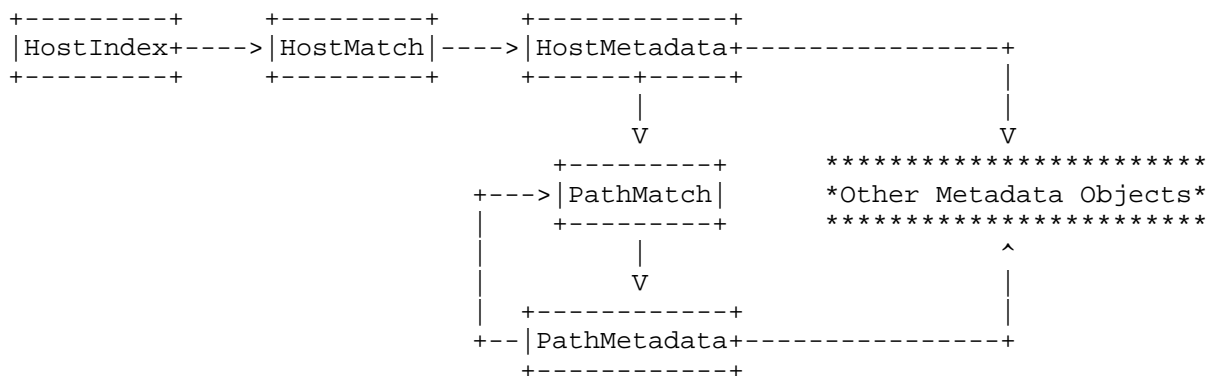
order to deliver the requested content.

The HostIndex links end-user facing Hostnames to HostMetadata objects, which contain (or reference) the default CDNI Metadata required to serve content for that Hostname. When looking up CDNI Metadata, the downstream CDN looks up the requested Hostname in the HostIndex, from there it can find HostMetadata which describes delivery rules for a Hostname and PathMetadata which may override those rules for given URI paths within the Hostname.

As well as containing the default CDNI Metadata for the specified Hostname, HostMetadata and PathMetadata objects may also contain PathMatch objects which in turn contain PathMetadata objects. PathMatch objects override the CDNI Metadata in the HostMetadata object or one or more preceding PathMetadata objects with more specific CDNI Metadata that applies to content requests matching the pattern defined in that PathMatch object.

For the purposes of retrieving CDNI Metadata all other required CDNI Metadata objects and their properties are discoverable from the appropriate HostMetadata, PathMatch and PathMetadata objects for the requested content.

The relationships between the HostIndex, HostMatch, HostMetadata, PathMatch and PathMetadata objects are described in Figure 1.



Key: ----> = References

Figure 1: Relationships between the HostIndex, HostMetadata & PathMetadata CDNI Metadata Objects

The table below describes the HostIndex, HostMetadata and PathMetadata objects in more detail.

Data Object	Description
HostIndex	A HostIndex object lists the Hostnames that an upstream CDN can provide CDNI metadata for and the URIs to use for retrieving that CDNI Metadata. For example, if "example.com" is a content provider, the HostIndex object may include an entry for "example.com" with the URI of the associated HostMetadata object. These hostnames are contained inside a list of HostMatch objects.
HostMatch	A HostMatch object defines a hostname to match against a requested host, and contains or references a HostMetadata object which contains CDNI Metadata properties to be applied when a content request matches against the hostname.
HostMetadata	A HostMetadata object contains (or references) the default CDNI Metadata properties for content served from that hostname, i.e. the CDNI Metadata properties for content requests that do not match any of the PathMatch objects contained or referenced by that HostMetadata object. For example, a HostMetadata object may describe the metadata properties which apply to "example.com" and may contain PathMatches for "example.com/movies/*" and "example.com/music/*" which reference corresponding PathMetadata objects that contain the CDNI Metadata properties for those specific URI paths.
PathMatch	A PathMatch object defines a pattern to match against the requested path, and contains or references a PathMetadata object which contains (or references) the CDNI Metadata properties to be applied when a content request matches against the defined URI path pattern.

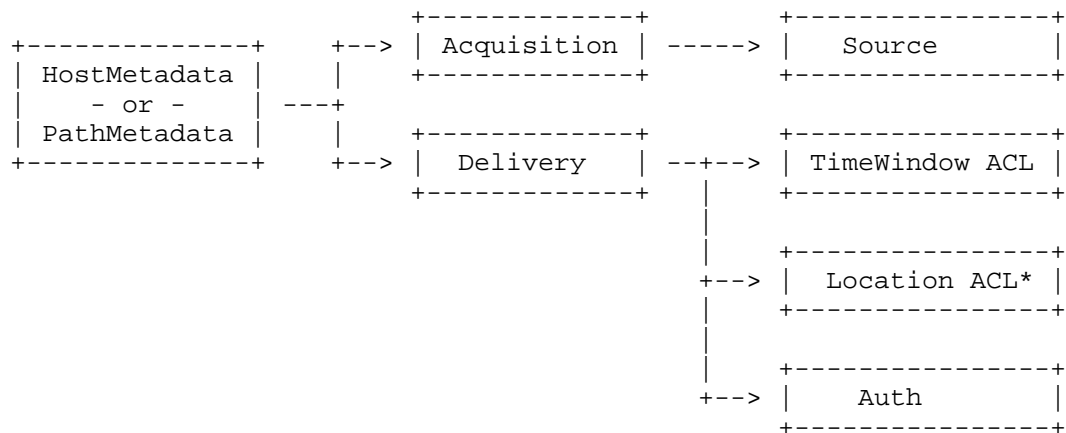
PathMetadata	A PathMetadata object contains the CDNI Metadata properties for content served with the associated URI path (defined in a PathMatch object). A PathMetadata object may also contain PathMatch objects in order to recursively define more specific URI paths that require different (e.g. more specific) CDNI Metadata to this one. For example, the PathMetadata object which applies to "example.com/movies/*" may describe CDNI metadata which apply to that resource path and may contain a PathMatch object for "example.com/movies/hd/*" which would reference the corresponding PathMetadata object for the "example.com/movies/hd/" path prefix.
--------------	--

Table 1: HostIndex, HostMetadata and PathMetadata CDNI Metadata Objects

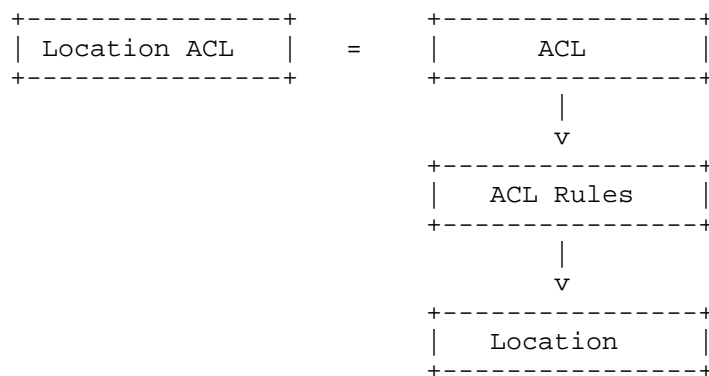
3.2. Remaining CDNI Metadata objects

The HostMetadata and PathMetadata objects contain or can reference other CDNI Metadata objects that contain properties which describe how User Agent requests for content should be processed, for example where to acquire the content, authorization rules that should be applied, delivery location restrictions and so on. The properties associated with the processing of User Agent requests fall into two categories, Delivery and Acquisition. Delivery properties, such as Location based restrictions, are contained or referenced within a Delivery object. Acquisition properties, such as which Origin Server to use to acquire the content, are contained or referenced within an Acquisition Object. Delivery and Acquisition objects contain or reference other CDNI Metadata objects to define the properties and rules which should be applied when processing requests for content. In some cases the rules that should be applied are complex but also likely to be reusable and repeated across many HostMetadata or PathMetadata objects.

The relationships between the HostMetadata and PathMetadata objects and the other CDNI Metadata objects (Delivery object, Acquisition object, etc.) required for CDNI request routing and delivery are illustrated in Figure 2.



*example ACL



Key: ----> = References

Figure 2: Relationships between HostMetadata and PathMetadata and the other CDNI Metadata Objects

The table below describes the remaining CDNI Metadata objects that were not defined in Section 3.1.

Data Object	Description
Acquisition	Container object for metadata that applies to content acquisition.
Delivery	Container object for metadata that applies to content delivery.
Source	Information needed by a dCDN to acquire content. For example the host to contact, the protocol to use for acquisition and any authentication and authorization methods that should be used.
ACL	Contains or references a list of ACLRules that are used to define any delivery restrictions that must be applied e.g. Location restrictions or Time based restrictions.
ACLRule	Contains or references a list of objects which define to what the restrictions should be applied e.g. an ACLRule may reference a Location Object if a location based ACL is required.
TimeWindow	Start and end time used to specify windows of availability or unavailability for the content.
Location	Geographic or network location identified by country code, BGP AS number, or subnet to which content may (or may not) be delivered.
Auth	Method and credentials for authentication and authorization including URI-signing, token-base, etc.

Table 2: Content Distribution Metadata Data Objects

The relationships in Figure 1 and Figure 2 are summarised in Table 3 below and the properties of each object are described in Section 4.1.

Data Object	Objects it References
HostIndex	0 or more HostMetadata objects.
HostMetadata	0 or more PathMatch objects. 0 or 1 Delivery objects. 0 or 1 Acquisition objects.
PathMatch	1 PathMetadata object.
PathMetadata	0 or more PathMatch objects. 0 or 1 Delivery objects. 0 or 1 Acquisition objects.
Acquisition	0 or more Source objects.
Delivery	0 or more ACL objects. 0 or more Auth objects.
ACL	0 or more ACLRule objects.

ACLRule	0 or more Location objects or 0 or more TimeWindow	
	objects.	
+-----+-----+-----+-----+-----+-----+		

Table 3: Relationships between CDNI Metadata Objects

3.3. Metadata Inheritance

In the data model, a HostMetadata object may contain (or reference) multiple PathMetadata objects. Each PathMetadata object may in turn contain (or reference) other PathMetadata objects. These relationships form a tree.

The tree of HostMetadata objects and PathMetadata objects forms an inheritance tree. Each node in the tree inherits the property values set by its parent.

In the tree, a child may override any property value which has been set by its parent. If a HostMetadata object sets the value of a property, that value may be overridden by a PathMetadata object (the child of the HostMetadata object). If a PathMetadata object contains (or references) other PathMetadata objects as children, then those children PathMetadata objects may override the property values set by the parent PathMetadata object.

If a child node overrides the value of a list, then the entire list is replaced with the value set by the child node. If a child node overrides the value of an object, then the whole object is replaced with the value set by the child node.

4. Encoding-Independent CDNI Metadata Object Descriptions

This section provides the definitions of each object type declared in Section 3. The definition of each object contains an unordered set of properties. The type of some properties is another CDNI Metadata object and in those cases the value of the property can be either an object of that type (the object is embedded) or a Link object that describes a URI and relationship that can be dereferenced to retrieve the CDNI Metadata object that should be used as the value of that property.

4.1. CDNI Metadata Data Object Descriptions

Each of the sub-sections below describes the properties associated with the data objects defined in Table 2.

4.1.1. HostIndex

The HostIndex object is the entry point into the CDNI Metadata hierarchy. An incoming content request is matched against the list of hosts to find the HostMatch object which applies to the request.

Property: hosts
Description: List of HostMatch objects
Type: List of HostMatch
Mandatory: Yes.

4.1.2. HostMatch

The HostMatch object contains a hostname to match against and a metadata object to apply if a match is found.

Property: hostname
Description: String to match against the requested host.
Type: String
Mandatory: Yes
Property: hostmetadata
Description: CDNI Metadata to apply when delivering content that matches this pattern.
Type: HostMetadata
Mandatory: Yes

4.1.3. HostMetadata

The HostMetadata object contains both metadata that applies to content requests for a particular host and a list of pattern matches for finding more specific metadata based on the resource path in a content request.

Property: acquisition
Description: Container for content acquisition related metadata.
Type: Acquisition
Mandatory: No. No default.
Property: delivery
Description: Container for content delivery related metadata.
Type: Delivery
Mandatory: No. No default.
Property: paths
Description: Path specific rules. First match applies.
Type: List of PathMatch
Mandatory: No. Default apply the properties defined in this HostMetadata object to all paths.

Property: hostname

Description: The end-user facing Hostname for this HostMetadata object.

Type: Hostname

Mandatory: Yes.

4.1.4. Acquisition

Metadata which provides the dCDN information about content acquisition e.g. how to contact an uCDN Surrogate or an Origin Server. The sources are not necessarily the actual Origin Servers operated by the CSP but might be a set of Surrogates in the uCDN.

Property: sources

Description: Sources from which the dCDN can acquire content.

Type: List of Source

Mandatory: No. Defaults to empty list.

4.1.5. Delivery

Metadata related to content delivery, e.g. delivery restrictions or content authorization methods.

Property: locations

Description: Access control list which applies restrictions to delivery based on client location.

Type: ACL

Mandatory: No. Defaults is allow all locations.

Property: times

Description: Access control list which applies restrictions to delivery based on request time.

Type: ACL

Mandatory: No. Defaults is allow all times.

Property: auth

Description: Options for authenticating content requests. All options in the list are equally valid.

Type: List of Auth

Mandatory: No. Defaults is no auth.

Property: protocol

Description: The delivery protocol to be used for content requests that match this HostMetadata object.

Type: protocol

Mandatory: Yes.

Property: active

Description: Enable or disable delivery from this host.

Type: boolean

Mandatory: No. Default yes.

4.1.6. PathMatch

The PathMatch object contains an expression to match against and a metadata object to apply if a match is found.

Property: pattern

Description: String to match against the requested path, i.e. against the [RFC3986] path-absolute.

Type: Pattern

Mandatory: Yes

Property: patternflags

Description: Flags to control the pattern match.

Type: List of PatternFlags

Mandatory: No. Default Case-sensitive infix matching.

Property: pathmetadata

Description: CDNI Metadata to apply when delivering content that matches this pattern.

Type: PathMetadata

Mandatory: Yes

4.1.7. PathMetadata

A PathMetadata object contains the CDNI Metadata properties for content served with the associated URI path (defined in a PathMatch object). Note that if CDNI metadata is used as an input to CDNI request routing and DNS-based redirection is employed, then any metadata at the PathMetadata level or below will be inaccessible at request routing time.

PathMetadata objects may contain any of the properties of a HostMetadata object with the following exceptions:

- o PathMetadata objects MUST NOT contain a hostname property.
- o PathMetadata objects MUST NOT contain a protocol property.
- o The presence of an sources property is OPTIONAL.

4.1.8. ACL

An ACL object contains or references a list of ACLRule objects which define a set of restrictions to apply to content delivery e.g. Location restrictions. An ACL may reference or contain ACLRules referencing or containing Location or TimeWindow objects but not both.

Property: aclrules
Description:
Type: List of ACLRule
Mandatory: No. Default no rules.

4.1.9. ACLRule

An ACLRule contains or references a list of either TimeWindow or Location objects. ACLRule objects are used to construct ACL to apply restrictions to content delivery.

Note: Although both the allow and deny properties are optional, one and only one of them MUST be present in an ACLRule. An ACLRule must also only refer to one of Location or TimeWindow but not both and should only refer to the objects relevant to the ACL type as defined by Delivery Metadata i.e. a Delivery Metadata object with an ACL with relationship of LocationACL must not reference TimeWindow objects further down in the Metadata hierarchy.

Property: allow
Description: List of either Locations (Location ACL) or Time Windows (TimeWindow ACL) which must be allowed.
Type: List of Location or TimeWindow
Mandatory: No. Default implicit Allow.

Property: deny
Description: List of either Locations (Location ACL) or Time Windows (TimeWindow ACL) which must be denied.
Type: List of Location or TimeWindow
Mandatory: No. Default implicit Deny.

4.1.10. TimeWindow

A TimeWindow object describes a time range which may be applied by an ACLRule, e.g. Start 09:00AM 01/01/2000 End 17:00PM 01/01/2000.

Property: start
Description: The start time of the window.
Type: Time
Mandatory: Yes

Property: end
Description: The end time of the window.
Type: Time
Mandatory: Yes

4.1.11. Location

A Location object describes a Location which may be applied by an ACLRule, e.g. a Location may be an IPv4 address range or a geographic location.

Property: iprange
Description: A set of IP Addresses.
Type: List of IPRange.
Mandatory: Yes

[Ed: Location as specified above only supports the Class 1a names described in [I-D.jenkins-cdni-names]. Need to add support for Class 1b names to a later version.]

4.1.12. Source

A Source object describes the Source which should be used by the dCDN for content acquisition, e.g. a Surrogate within the uCDN or an alternate Origin Server, the protocol to be used and any authentication method.

Property: auth
Description: Authentication method to use when requesting content from this source.
Type: Auth
Mandatory: No. Default is no authentication.
Property: endpoints
Description: Origins from which the dCDN can acquire content.
Type: List of EndPoint
Mandatory: Yes.
Property: protocol
Description: Protocol to use for content acquisition.
Type: Protocol
Mandatory: Yes.

4.1.13. Auth

An Auth object defines authentication and authorization methods to be used during content delivery and content acquisition, e.g. methods such as tokenization and URL Signing.

Property: type
Description: A string containing the authentication type "url-signing", "url-token", "http-basic", or "http-digest". The type dictates which optional fields are present and valid in the rest of the object. The "url-signing" type refers to URL signing authentication. The "url-token" type refers to token-

based authentication. The "basic" and "digest" types refer to HTTP Basic and Digest access authentication.

Type: String

Mandatory: Yes.

Property: algo

Description: A string containing the signature algorithm (e.g. "md5", "sha-1", etc.).

Type: String

Mandatory: Yes, if type is "url-signing".

Property: symmetric

Description: A boolean if true, URL signing uses symmetric keys, otherwise asymmetric.

Type: boolean

Mandatory: Yes, if type is "url-signing".

Property: key

Description: A hex-encoded number containing the public key for verifying signatures, only valid if "symmetric" field is set to false.

Type: boolean

Mandatory: Yes, if type is "url-signing".

Property: username

Description: A string containing the username for "basic" and "digest" types.

Type: String

Mandatory: Yes, if type is "basic" or "digest".

Property: password

Description: A string containing the password for "basic" and "digest" types.

Type: String

Mandatory: Yes, if type is "basic" or "digest".

4.1.14. Link

A link object may be used in place of any of the objects described above. Links can be used to avoid duplication if the same metadata information is repeated within the metadata tree. When a link replaces an object, its href property is set to the URI of the resource, its rel property is set to the name of the property it is replacing, and its type property is set to the type of the object it is replacing.

Property: href

Description: The URI of the of the addressable object being referenced.

Type: URI

Mandatory: Yes

Property: rel
Description: The Relationship between the referring object and the object it is referencing.
Type: String
Mandatory: Yes
Property: type
Description: The type of the object being referenced.
Type: String
Mandatory: Yes

4.2. CDNI Metadata Simple Data Type Descriptions

This section describes the simpler data types that are used for properties of CDNI Metadata objects.

4.2.1. Protocol

This type only appears in Links. Links with this type are not machine readable but rather represent particular feature sets of a protocol defined in a specification and implemented in code. The URI contained in the link needs to be defined for each delivery protocol with an associated interoperable feature set.

The following examples are illustrative:

- o http://url.cdni.ietf.example/protocol/delivery/http/rfcABCD
- o http://url.cdni.ietf.example/protocol/delivery/rtmp/rfcEFGH
- o http://url.vendorY.ietf.example/protocol/delivery/rtmp/releaseP.Q

[Editor's Note: It may be more appropriate to use the 'tag' URI scheme [RFC4151] for these URIs.]

4.2.2. Endpoint

A hostname (with optional port) or an IP address (with optional port).

Note: Client implementations MUST support IPv4 addresses encoded as specified by the 'IPv4address' rule in Section 3.2.2 of [RFC3986] and MUST support all IPv6 address formats specified in [RFC4291]. Server implementations SHOULD use IPv6 address formats specified in [RFC5952].

4.2.3. IPRange

One of:

- o A range of consecutive IP addresses (IPv4 or IPv6) expressed as Address1-Address2 which does not have to be to power of two aligned, for example the range 192.0.2.1-192.0.2.10 is valid. The first Address in the range MUST be 'lower' than the final address in the range.
- o A valid IP subnet (IPv4 or IPv6) expressed using CIDR notation.
- o A single IP address (IPv4 or IPv6).

Note: Client implementations MUST support IPv4 addresses encoded as specified by the 'IPv4address' rule in Section 3.2.2 of [RFC3986] and MUST support all IPv6 address formats specified in [RFC4291]. Server implementations SHOULD use IPv6 address formats specified in [RFC5952].

4.2.4. Pattern

A pattern for string matching paths. The string may contain the wildcards * and ?.

- o * matches any sequence of characters (including the empty string).
- o ? matches exactly one character.

Escaping: The three literals \ , * and ? should be escaped as \\, * and \?

4.2.5. PatternFlags

A set of flags indicating how a pattern match is made. The flags are:

- o Case-insensitive - Perform a case insensitive match (absence indicates case-sensitive match).
- o Prefix - Match against the start of the string (absence indicates that a match may start anywhere in the string).
- o Suffix - Match against the end of the string (absence indicates that a match may end anywhere in the string).

Absence of both Prefix and Suffix results in a match against any part of the string (infix).

4.2.6. URI

A URI as specified in [RFC3986].

4.2.7. Time

A time value expressed in seconds since Unix epoch in the UTC timezone.

5. CDNI Metadata interface

This section specifies an interface to enable a Downstream CDN to retrieve CDNI Metadata objects from an Upstream CDN.

The interface can be used by a Downstream CDN to retrieve CDNI Metadata objects either dynamically as required by the Downstream CDN to process received requests (for example in response to receiving a CDNI Request Routing request from an Upstream CDN or in response to receiving a request for content from a User Agent) or in advance of being required.

The CDNI Metadata interface is built on the principles of RESTful web services. This means that requests and responses over the interface are built around the transfer of representations of hyperlinked resources. A resource in the context of the CDNI Metadata interface is any object in the Data Model (as described in Section 3 through Section 4.1).

In the general case a CDNI Metadata server makes each instance of an addressable CDNI Metadata object available via a unique URI that returns a representation of that instance of that CDNI Metadata object. When an object needs to reference another addressable CDNI Metadata object (for example a HostIndex object referencing a HostMetadata object) it does so by including a link to the referenced object.

CDNI Metadata servers are free to assign whatever structure they desire to the URIs for CDNI Metadata objects and CDNI Metadata clients MUST NOT make any assumptions regarding the structure of CDNI Metadata URIs or the mapping between CDNI Metadata objects and their associated URIs. Therefore any URIs present in the examples below are purely illustrative and are not intended impose a definitive structure on CDNI Metadata interface implementations.

5.1. Transport

The CDNI Metadata interface uses HTTP as the underlying protocol transport.

The HTTP Method in the request defines the operation the request would like to perform. Servers implementing the CDNI Metadata interface MUST support the HTTP GET and HEAD methods.

The corresponding HTTP Response returns the status of the operation in the HTTP Status Code and returns the current representation of the resource (if appropriate) in the Response Body. HTTP Responses from servers implementing the CDNI Metadata interface that contain a

response body SHOULD include an ETag to enable validation of cached versions of returned resources.

The CDNI Metadata interface specified in this document is a read-only interface. Therefore support for other HTTP methods such as PUT, POST and DELETE etc. is not specified. Server implementations of this interface SHOULD reject all methods other than GET and HEAD.

As the CDNI Metadata interface builds on top of HTTP, CDNI Metadata servers may make use of any HTTP feature when implementing the CDNI Metadata interface, for example a CDNI Metadata server may make use of HTTP's caching mechanisms to indicate that the returned response/representation can be reused without re-contacting the CDNI Metadata server.

5.2. Retrieval of CDNI Metadata resources

In the general case a CDNI Metadata server makes each instance of an addressable CDNI Metadata object available via a unique URI and therefore in order to retrieve CDNI Metadata, a CDNI Metadata client first makes a HTTP GET request for the URI of the HostIndex which provides the CDNI Metadata client with a list of Hosts (along with their public facing hostnames) that the upstream CDN may delegate to the downstream CDN.

In order to retrieve the CDNI Metadata for a particular request the CDNI Metadata client processes the received HostIndex object and finds the corresponding HostMetadata entry (by matching the hostname in the request against the hostnames in the HostIndex). The CDNI metadata client then makes a GET request for the URI specified in the href key of that Host's entry in the HostIndex.

In order to retrieve the most specific metadata for a particular request, the CDNI metadata client inspects the HostMetadata for references to more specific PathMetadata objects. If any PathMetadata match the request, the CDNI metadata client makes another GET request for the PathMetadata. Each PathMetadata object may also include references to yet more specific metadata. If this is the case, the CDNI metadata client continues requesting PathMetadata recursively.

Where a downstream CDN is interconnected with multiple upstream CDNs, the downstream CDN must decide which upstream CDN's metadata should handle a particular User Agent request.

In the case of where application level redirection (e.g. HTTP 302 redirects) is being used between CDNs, it is expected that the downstream CDN will be able to determine the upstream CDN that

redirected a particular request from information contained in the received request (e.g. via the URI in case of HTTP redirection across CDNs). With knowledge of which upstream CDN routed the request, the downstream CDN can choose the correct metadata server.

In the case of DNS redirection there is not sufficient information carried in the DNS request from User Agents to determine the upstream CDN that redirected a particular request and therefore downstream CDNs may have to apply local policy when deciding which upstream CDN's metadata to apply.

5.3. Bootstrapping

The URI for the HostIndex object of a given upstream CDN needs to be either discovered by or configured in the downstream CDN. All other objects/resources are then discoverable from the HostIndex object by following the links in the HostIndex object and the referenced HostMetadata and PathMetadata objects.

If the URI for the HostIndex object is not manually configured in the downstream CDN then the HostIndex URI could be discovered via the CDNI Control interface. An upstream CDN would advertise the URI of the HostIndex object to the downstream CDN via the CDNI Control Interface.

5.4. Encoding

Object are resources that may be:

- o Addressable, where the object is a resource that may be retrieved or referenced via its own URI.
- o Embedded, where the object is contained (or inlined) within a property of an addressable object.

In the descriptions of objects we use the term "X contains Y" to mean either Y is directly embedded in X or that Y is linked to by X. It is generally a deployment choice for the uCDN implementation to decide when and which CDNI Metadata objects to embed and which are separately addressable.

5.4.1. MIME Media Types

All MIME types are prefixed with "application/cdni." The MIME type for each object matches the type name of that object as defined by this document. Table 4 lists a few examples of the MIME Media Type for each object (resource) that is retrievable through the CDNI Metadata interface. The MIME type suffix depends on the metadata encoding, either "+xml" or "+json".

Data Object	MIME Media Type
HostIndex	application/cdni.HostIndex
HostMatch	application/cdni.HostMatch
HostMetadata	application/cdni.HostMetadata
PathMatch	application/cdni.PathMatch
PathMetadata	application/cdni.PathMetadata

Table 4: MIME Media Types for CDNI Metadata resources

See <http://www.iana.org/assignments/media-types/index.html> for reference.

5.4.2. JSON Encoding of Objects

One possible encoding for a CDNI Metadata object is a JSON object containing a dictionary of (key,value) pairs where the keys are the property names and the values are the associated property values.

The keys of the dictionary are the names of the properties associated with the object and are therefore dependent on the specific object being encoded (i.e. dependent on the MIME Media Type of the returned resource). Likewise, the values associated with each key are dependent on the specific object being encoded (i.e. dependent on the MIME Media Type of the returned resource).

Dictionary keys in JSON are case sensitive and therefore any dictionary key defined by this document (for example the names of CDNI Metadata object properties) MUST always be represented in lowercase.

In addition to the properties of the object, the following three additional keys defined below may be present in any object.

Key: base

Description: Provides a prefix for any relative URLs in the object. This is similar to the XML base tag [XML-BASE]. If absent, all URLs in the remainder of the document must be absolute URLs.

Type: URI

Mandatory: No

Key: links

Description: The links of this object to other addressable objects. Any property may be replaced by a link to an object with the same type as the property it replaces.

Type: List of Link
Mandatory: Yes

5.4.2.1. JSON Example

A downstream CDN may request the HostIndex and receive the following object of type "application/cdni.HostIndex+json":

```
{
  "host": [
    {
      "hostname": "video.example.com",
      "links": [
        {
          "rel": "hostmetadata",
          "type": "HostMetadata",
          "href": "http://metadata.ucdn.com/video"
        }
      ]
    },
    {
      "hostname": "images.example.com",
      "links": [
        {
          "rel": "hostmetadata",
          "type": "HostMetadata",
          "href": "http://metadata.ucdn.com/images"
        }
      ]
    }
  ]
}
```

If the incoming request has a Host header with "video.example.com" then the downstream CDN would fetch from the next metadata object from "http://metadata.ucdn.com/video" expecting a MIME type of "application/cdni.HostMetadata+json":

```
{
  "hostname": "video.example.com",
  "acquisition": {
    "source": [
      {
        "links": [{
          "rel": "auth",
          "type": "Auth",
          "href": "http://metadata.ucdn.com/auth1234"
        }],
      }
    ]
  }
}
```

```
        "endpoint": "acq1.ucdn.com",
        "protocol": "ftp"
    },
    {
        "links": [{
            "rel": "auth",
            "type": "Auth",
            "href": "http://metadata.ucdn.com/auth1234"
        }],
        "endpoint": "acq2.ucdn.com",
        "protocol": "http"
    }
]
},
"delivery": {
    "location": {
        "aclrule": {
            "deny": { "iprange": "192.168.0.0/16" }
        }
    },
    "auth": {

    },
    "protocol": "http",
    "active": "true"
},
"path": [
    {
        "pattern": "/videos/trailers/*",
        "patternflags": "prefix",
        "links": [{
            "rel": "pathmetadata",
            "type": "PathMetadata",
            "href": "http://metadata.ucdn.com/videos/trailers"
        }]
    },
    {
        "pattern": "/videos/movies/*",
        "patternflags": "prefix",
        "links": [{
            "rel": "pathmetadata",
            "type": "PathMetadata",
            "href": "http://metadata.ucdn.com/videos/movies"
        }]
    }
]
}
```

Suppose the path of the requested resource matches the `"/video/movies/*"` pattern, the next metadata requested would be for `"http://metadata.ucdn.com/video/movies"` with an expected type of `"application/cdni.PathMetadata"`:

```
{
  "delivery": {
    "auth": {

    }
  },
  "path": {
    "pattern": "/videos/movies/hd/*",
    "patternflags": "prefix",
    "links": [{
      "rel": "pathmetadata",
      "type": "PathMetadata",
      "href": "http://metadata.ucdn.com/videos/movies/hd"
    }]
  }
}
```

Finally, if the path of the requested resource also matches the `"/videos/movies/hd/*"` pattern, the downstream CDN would also fetch the following object from `"http://metadata.ucdn.com/videos/movies/hd"` with MIME type `"application/cdni.PathMetadata"`:

```
{
  "delivery": {
    "time": {
      "aclrule": {
        "allow": {
          "start": "1213948800",
          "end": "1327393200"
        }
      }
    }
  }
}
```

5.4.3. XML Encoding of Objects

Another possible encoding for a CDNI Metadata object is an XML document containing elements with tag names which match property names and values which match the associated property values.

Tag names of elements are the names of the properties associated with the object and are therefore dependent on the specific object being

encoded (i.e. dependent on the MIME Media Type of the returned resource). Likewise, the values associated with each element are dependent on the specific object being encoded (i.e. dependent on the MIME Media Type of the returned resource).

Lists are encoded by repeating the singular form of a property name. For example the "hosts" property is a list of "HostMatch" objects. This list would be encoded as multiple "host" elements.

Link objects are a special case. If a Link object replaces a property then a "link" element replaces the expected element. The properties of the Link object are encoded as XML attributes. The type attribute is set to the MIME type of the target object. The href attribute is set to the URI of the target object. The rel attribute is set to the name of the element being replaced.

5.4.3.1. XML Example

A downstream CDN may request the HostIndex and receive the following object of type "application/cdni.HostIndex+json":

```
<HostIndex>
  <host>
    <hostname>video.example.com</hostname>
    <link rel="hostmetadata" type="HostMetadata"
      href="http://metadata.ucdn.com/video"/>
  </host>
  <host>
    <hostname>images.example.com</hostname>
    <link rel="hostmetadata" type="HostMetadata"
      href="http://metadata.ucdn.com/images"/>
  </host>
</HostIndex>
```

If the incoming request has a Host header with "video.example.com" then the downstream CDN would fetch from the next metadata object from "http://metadata.ucdn.com/video" expecting a MIME type of "application/cdni.HostMetadata+json":

```
<HostMetadata>
  <hostname>video.example.com</hostname>
  <acquisition>
    <source>
      <link rel="auth" type="Auth"
        href="http://metadata.ucdn.com/auth1234"/>
      <endpoint>acq1.ucdn.com</endpoint>
      <protocol>ftp</protocol>
    </source>
    <source>
      <link rel="auth" type="Auth"
        href="http://metadata.ucdn.com/auth1234"/>
      <endpoint>acq2.ucdn.com</endpoint>
      <protocol>http</protocol>
    </source>
  </acquisition>
  <delivery>
    <location>
      <aclrule>
        <deny>
          <iprange>192.168.0.0/16</iprange>
        </deny>
      </aclrule>
    </location>
    <auth></auth>
    <protocol>http</protocol>
    <active>true</active>
  </delivery>
  <path>
    <pattern>/videos/trailers/*</pattern>
    <patternflags>prefix</patternflags>
    <link rel="pathmetadata" type="PathMetadata"
      href="http://metadata.ucdn.com/videos/trailers"/>
  </path>
  <path>
    <pattern>/videos/movies/*</pattern>
    <patternflags>prefix</patternflags>
    <link rel="pathmetadata" type="PathMetadata"
      href="http://metadata.ucdn.com/videos/movies"/>
  </path>
</HostMetadata>
```

Suppose the path of the requested resource matches the "/video/movies/*" pattern, the next metadata requested would be for "http://metadata.ucdn.com/video/movies" with an expected type of "application/cdni.PathMetadata":


```
<PathMetadata>
  <delivery>
    <auth></auth>
  </delivery>
  <path>
    <pattern>/videos/movies/hd/*</pattern>
    <patternflags>prefix</patternflags>
    <link rel="pathmetadata" type="PathMetadata"
      href="http://metadata.ucdn.com/videos/movies/hd"/>
  </path>
</PathMetadata>
```

Finally, if the path of the requested resource also matches the `"/videos/movies/hd/*"` pattern, the downstream CDN would also fetch the following object from `"http://metadata.ucdn.com/videos/movies/hd"` with MIME type `"application/cdni.PathMetadata"`:

```
<PathMetadata>
  <delivery>
    <time>
      <aclrule>
        <allow>
          <start>1213948800</start>
          <end>1327393200</end>
        </allow>
      </aclrule>
    </time>
  </delivery>
</PathMetadata>
```

5.5. Extensibility

The set of metadata properties may be extended with proprietary and / or custom properties. New properties may be added to any existing object.

The names of such properties MUST begin with an `"x-"` prefix. If a property is vendor specific, then `"x-vendor-"` SHOULD be used as the name prefix, where the `"vendor"` string is replaced by the name of the vendor.

The values of new properties MAY include an `"ignorable"` property with a boolean type. If `"ignorable"` is set to true, then request routers and surrogates in any interconnected CDN MAY safely ignore the new property. If `"ignorable"` is set to false, then a CDN which does not understand the property MUST NOT service a request for the corresponding content.

6. IANA Considerations

This document requests the registration of the "application/cdni" MIME type.

7. Security Considerations

The CDNI Metadata Interface is expected to be secured as a function of the transport protocol (e.g. HTTP authentication).

If a malicious metadata server is contacted by a downstream CDN, the malicious server may provide metadata to the downstream CDN which denies service for any piece of content to any user agent. The malicious server may also provide metadata which directs a downstream CDN to a malicious origin server instead of the actual origin server.

A malicious metadata client could request metadata for a piece of content from an upstream CDN. However, given the current set of metadata properties, no useful information would be compromised.

8. Acknowledgements

The authors would like to thank David Ferguson and Francois le Faucheur for their valuable comments and input to this document.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.

9.2. Informative References

- [I-D.davie-cdni-framework]
Davie, B. and L. Peterson, "Framework for CDN Interconnection", draft-davie-cdni-framework-00 (work in progress), July 2011.

- [I-D.ietf-cdni-problem-statement]
Niven-Jenkins, B., Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", draft-ietf-cdni-problem-statement-03 (work in progress), January 2012.
- [I-D.ietf-cdni-requirements]
Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", draft-ietf-cdni-requirements-02 (work in progress), December 2011.
- [I-D.zyp-json-schema]
Zyp, K. and G. Court, "A JSON Media Type for Describing the Structure and Meaning of JSON Documents", draft-zyp-json-schema-03 (work in progress), November 2010.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4151] Kindberg, T. and S. Hawke, "The 'tag' URI Scheme", RFC 4151, October 2005.
- [RFC4287] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", RFC 4287, December 2005.
- [XML-BASE]
Marsh, J., Ed. and R. Tobin, Ed., "XML Base (Second Edition) - <http://www.w3.org/TR/xmlbase/>", January 2009.

Appendix A. Relationship to the CDNI Requirements

Section 6 of [I-D.ietf-cdni-requirements] lists the requirements for the CDNI Metadata Distribution interface. This section outlines which of those requirements are met by the CDNI Metadata interface specified in this document.

All metadata requirements are met either directly or indirectly by the CDNI Metadata Interface described in this document. The following paragraphs describe notable exceptions.

Requirements related to pre-positioning of metadata are not met directly by this document. Triggering metadata pre-positioning is beyond the scope of the CDNI Metadata interface. However, the interface as described by this document supports pulling metadata on-

demand for the purpose of pre-positioning.

Requirement META-13 relating to feedback from the downstream CDN to the upstream CDN with respect to metadata is not directly supported by the pull-based interface described in this document. As an alternative, the downstream CDN may use the CDNI Logging interface to convey error conditions related to metadata.

Requirement META-18 relating to surrogate cache behavior parameters is supported via extensibility. However, the example parameters in META-18 are not described in this document.

Authors' Addresses

Ben Niven-Jenkins
Velocix (Alcatel-Lucent)
3 Ely Road
Milton, Cambridge CB24 6AA
UK

Email: ben@velocix.com

Rob Murray
Velocix (Alcatel-Lucent)
3 Ely Road
Milton, Cambridge CB24 6AA
UK

Email: rmurray@velocix.com

Grant Watson
Velocix (Alcatel-Lucent)
3 Ely Road
Milton, Cambridge CB24 6AA
UK

Email: gwatson@velocix.com

Matt Caulfield
Cisco Systems
1414 Massachusetts Avenue
Boxborough, MA 01719
USA

Phone: +1 978 936 9307
Email: mcaulfie@cisco.com

Kent Leung
Cisco Systems
3625 Cisco Way
San Jose 95134
USA

Phone: +1 408 526 5030
Email: kleung@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 21, 2012

Danhua. Wang, Ed.
Huawei Technologies
Xiaoyan. He
Spencer. Dawkins
Huawei
Chen. Ge
China Telecom
Wei. Ni
Yunfei. Zhang
China Mobile
July 16, 2012

Routing Request Redirection for CDN Interconnection
draft-he-cdni-routing-request-redirection-02

Abstract

The Request Routing Interface comprises of (1) the asynchronous advertisement of footprint and capabilities by a dCDN that allows a uCDN to decide whether to redirect particular user requests to that dCDN; and (2) the synchronous operation of an upstream CDN requesting whether a downstream CDN is prepared to accept a user request and of a downstream CDN responding with how to actually redirect the user request. This document describes an interface for the latter part.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 21, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Interface function and operation overview	4
3.1. Discussion on protocol type for CDNI RRI	5
4. Data passed in CDNI request routing requests & responses	7
4.1. Data passed in CDNI request routing requests	7
4.2. Data passed in CDNI request routing responses	8
5. HTTP based RESTful interface for CDNI Request Routing	9
6. Protocol Specification	11
6.1. Recursive Request Routing	11
6.1.1. DNS based Request Routing Protocol	11
6.1.2. HTTP based Request Routing Protocol	12
6.2. Iterative Request Routing	12
7. Security Considerations	13
8. IANA Considerations	13
9. References	13
9.1. Normative Reference	13
9.2. Informative Reference	13
Authors' Addresses	14

1. Introduction

A Content Delivery Network (CDN) is a system built on an existing IP network which is used for large scale content delivery, via prefetching or dynamically caching contents on its distributed surrogates (caching servers) that are typically deployed close to the end users so that a CDN can improve access to the content it caches, for example, by reducing access latency and improving an end user's experience.

In recent years the volume of video and multimedia content delivered over the internet is rapidly increasing. To accommodate this increase, existing CDN providers are scaling up their infrastructure and many Network Service Providers (NSPs) are deploying their own CDNs. Another emerging requirement is CDN Interconnection (CDNI).

Several real world use cases are described in [I-D.draft-cdni-use-cases] which prove the necessity for CDN interconnection. The most frequently mentioned use case is leveraging the collective CDN footprint of interconnected standalone CDNs to achieve the goal of delivering content to additional distributed end users regardless of their location.

[I-D.draft-cdni-problem-statement] describes the problem area, where CDNs are interconnected as described in [I-D.draft-cdni-use-cases] based on the requirements described in [I-D.draft-cdni-requirements], and using the technology framework described in [I-D.davie-cdni-framework].

The purpose of this document is to define the interface for synchronous redirection operation of the request routing interface, which is one of the main building blocks of the CDN interconnection architecture described in [I-D.draft-cdni-requirements].

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

This document reuses the terminology defined in [I-D.draft-cdni-problem-statement]. The term "Distinguished CDN Domain" defined in [I-D.davie-cdni-framework] is also reused in this document.

The following additional terms are introduced by this document:

DNS Redirection: The act of using DNS name resolution for the request

routing process of a CDN. In DNS Redirection, the DNS resolver of the CDN makes the routing decision based on a local policy and returns the result as the response of a DNS query request to redirect a user agent to a new target. In CDNI, the result may point to a surrogate of the CDN, a request router in a downstream CDN or a surrogate in a downstream CDN, etc.

HTTP Redirection: The act of using an HTTP redirection response to redirect a user agent to a new target. The new target is the result of the routing decision of a CDN at the time it receives a content request via HTTP. In CDNI, the result may point to a surrogate of the CDN, a request router in a downstream CDN or a surrogate in a downstream CDN etc.

3. Interface function and operation overview

Editor's note: How an upstream CDN decides which downstream CDN(s) to query is outside the scope of this document.

The CDNI Request Routing interface is one of the main building blocks required in order to interconnect CDNs. The main function of the Request Routing interface is to allow the Request Routing systems in interconnected CDNs to communicate to facilitate the redirection of User Agent requests between interconnected CDNs.

The detailed requirements for the CDNI Request Routing interface and their relative priorities of those requirements are described in section 5 of [I-D.draft-cdni-requirements].

The CDNI Request Routing interface operates between a pair of interconnected CDNs. To enable communication over the CDNI Request Routing Interface, the two interconnected CDNs need to know the end point (URI) in the other CDN to query. For example, an Upstream CDN needs to know the URI (end point) in a Downstream CDN to send its CDNI Request Routing queries to.

The CDNI Request Routing URI may be statically pre-configured, dynamically discovered via the CDNI control interface, or discovered via other means. However, such discovery mechanisms are not specified in this document, as they are considered out of the scope of the CDNI Request Routing Protocol.

The CDNI Request Routing interface must support both of the request routing mechanisms illustrated in section 3.2 and 3.4 of [I-D.davie-cdni-framework], namely Iterative Request Routing and Recursive Request Routing. The Iterative Request Routing method does not invoke any interaction over the request routing interface across

interconnected CDNs. This document hence will not discuss Iterative Request Routing further.

In the case of Recursive Request Routing, an Upstream CDN forwards a routing request from a user agent to a Downstream CDN for surrogate selection. The initial candidate protocols for these interactions are DNS and HTTP. Moreover, the inside routing mechanisms used between the CDN and the user agent (DNS and HTTP Redirection) of the two interconnected CDNs should also be taken into account as they may affect the type of query request the Upstream CDN send to a Downstream CDN and the information the Downstream CDN return in its query response.

3.1. Discussion on protocol type for CDNI RRI

The request routing process has several variants depending on the factors including:

- Which routing mechanism is adopted by an Upstream CDN inside, DNS Redirection or HTTP Redirection.

- Which protocol is adopted for the CDNI Request Routing Interface.

- Which routing mechanism is adopted by a Downstream CDN inside, DNS Redirection or HTTP Redirection.

All possible combinations and their validity are shown in Table 1.

CaseNO.	uCDN Received Request	RRI Interface	dCDN Response	Note
1	DNS	DNS based	DNS with IP address of RR	dCDN works in HTTP Redirection mode, illustrated in section 3.1.1.1.
2	DNS	DNS based	DNS with hostname of RR	dCDN works in DNS Redirection mode, illustrated in section 3.1.1.2.
3	DNS	HTTP based	Invalid case	Protocol conversion occurs in uCDN, invalid case.
4	HTTP	HTTP based	HTTP 302 Redirection	dCDN works in HTTP Redirection mode, illustrated in section 3.1.2.
5	HTTP	HTTP based	DNS Redirection	dCDN works in DNS Redirection mode, invalid case.
6	HTTP	DNS based	Invalid case	Protocol conversion occurs, invalid case.

Table 1: Recursive Routing Cases

The rules to filter the cases and determine the validity of them are discussed below.

The Upstream CDN must not perform protocol conversion (A DNS query to an HTTP request or vice versa). To assist the routing decision of a Downstream CDN, the Upstream CDN shall convey as much information as possible to the Downstream CDN, e.g. URI of the requested content, the client's location information. In the case of HTTP to DNS conversion, a DNS request cannot convey all the information an HTTP request contains. In the case of DNS to HTTP conversion, a full HTTP URL cannot be constructed through a simple domain name contained by a DNS query request. Hence it is concluded that the protocol type used in the CDNI Request Routing Interface will be consistent with the one the Upstream CDN received from the user agent. Case3, Case6 are invalid according to this rule.

The Downstream CDN can determine according to its local policy a DNS Redirection or an HTTP Redirection to be adopted. When receiving a DNS query request over the CDNI Request Routing Interface.

If DNS Redirection is selected, as the location information has been changed to the Upstream CDN's when it proxies the DNS query request, the Downstream CDN cannot get the user agent's location information from the query request. The Downstream CDN sends a response with a CNAME of the hostname of the Request Router, so that the user agent issues another DNS query request which will convey its location information as shown in case2. If HTTP Redirection is selected, the Downstream CDN sends a response with the IP address of its Request Router, so that it can receive a subsequent content request based on HTTP containing the client's location information, to allow selection of an appropriate surrogate as shown in case1.

Based on filter rules above, Case 1, 2, and 4 are valid cases for CDNI. The following section describes these cases in detail.

4. Data passed in CDNI request routing requests & responses

As to the data that have to be passed between the CDNI request routing requests and responses, we believe that besides End User's request/response, certain CDNI metadata or policies might also be required. For example, in the CDNI request routing requests, appropriate CDNI Metadata could be introduced to aid the downstream CDN in making its decision. In the response, some policies might be included and returned to the End User to indicate how the responses could be reused, e.g., "the same response can be used without asking me again provided the End User is in this IP Address range", etc. As follows, detailed description of data that should be passed in the CDNI request routing requests and responses will be given respectively, considering under both DNS redirection and HTTP redirection mechanisms.

4.1. Data passed in CDNI request routing requests

The data passed in CDNI request routing requests splits into two basic categories, an encapsulation of the User Agent's request to the upstream CDN; and properties/parameters that the uCDN can use to control the dCDN's response or that can help the dCDN make its decision. Data passed in CDNI request routing requests are enumerated in Table 1, including both DNS request and HTTP request.

Table1. Data Passed in CDNI Request Routing Requests

	DNS Request	HTTP Request
An encapsulation of the User Agent's request	DNS Client IP address	Client IP address
	Type of query	Requested URL
	Real Client IP address(if know)	
	URI of requested content	
Properties that uCDN can use to control the dCDN's response	A link to the associated CDNI Metadata	A link to the associated CDNI Metadata
		Whether to return hostnames or IP address in the redirection URL

Note: As we presented before, the downstream CDN needs to obtain the appropriate CDNI Metadata to know how to process the CDNI Request Routing requests. We do not include actual CDNI Metadata in the CDNI Request Routing requests, only a link to the CDNI Metadata is included, shown in Table 1.

4.2. Data passed in CDNI request routing responses

There are also two basic categories of data passed in CDNI request routing responses, an encapsulation of the DNS response or HTTP response to return to the End User; parameters /policies that indicate the properties of the response, such as, whether it is cacheable, the scope of the response, etc. Data passed in CDNI request routing responses are enumerated in Table 2, including both DNS response and HTTP response.

Table2. Data Passed in CDNI Request Routing Responses

	DNS Response	HTTP Response
An encapsulation of CDNI/HTTP response to the End User	CNAME of the dCDN's Request Router	IP address or hostname of request router
	Parameter that indicate whether the response is cacheable	
Parameters/policies that indicate the properties of the response	Parameter that indicate how long to reduce the number of subsequent CDNI request routing requests the uCDN needs to make	
	Parameter that indicate the scope of the response (if it is cacheable), e.g., does it apply to a wider range of client IP addresses or URIs than one in the request	

5. HTTP based RESTful interface for CDNI Request Routing

This document defines a simple HTTP based RESTful interface for CDNI Request Routing, where End User's requests are encapsulated along with links to appropriate CDNI Metadata resources (records) and any other data that can aid the downstream CDN in processing the requests. The response encapsulates the response that the upstream CDN should return to the End User (if it decides to utilize the Downstream CDN for delivery) along with the policy for how the response could be reused (through standard HTTP Cache-Control headers).

The same RESTful interface is used for both DNS and HTTP redirection of User Agent's requests, although the contents of the CDNI Request Routing Interface requests/responses contain data specific to either DNS or HTTP redirection.

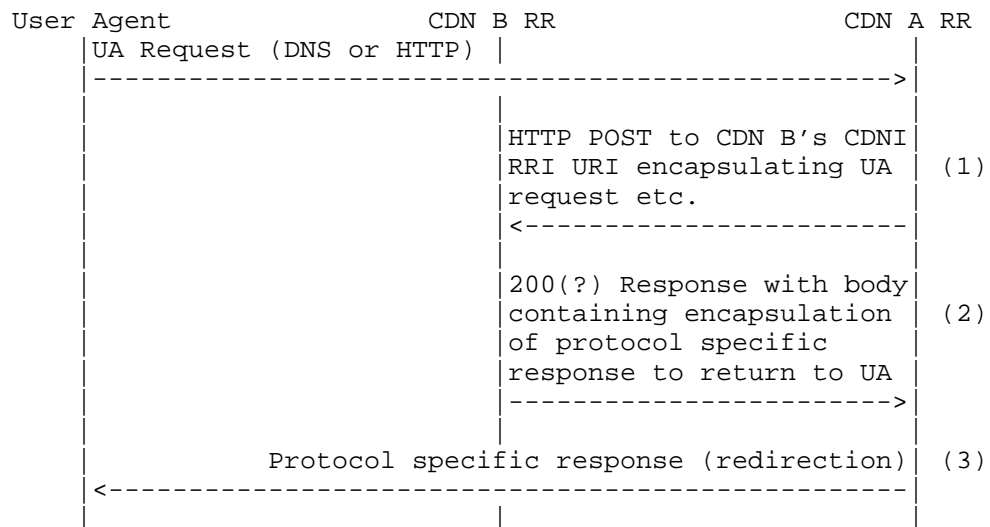
This approach has been chosen because it enables CDN operators to only have to deploy a single (RESTful) interface for request routing between their CDNs, regardless of the User Agent redirection method. In this way, from an operational point of view there is only one interface to monitor, manage, or develop troubleshooting tools for.

In addition, having a single CDNI Request Routing interface where the User Agent's DNS or HTTP request are encapsulated along with the

other data required for the downstream CDN to make a request routing decision, avoids having to try and encapsulate or proxy DNS/HTTP/RTMP/etc requests and find ways to somehow embed the additional CDNI Request Routing properties/data within those End User DNS/HTTP/RTMP/etc requests.

Finally, the interface is easily extendable to support other User Agent request redirection methods (e.g. RTMP 302 redirection).

The general call flow between Request Routers in a pair of interconnected CDNs is as follows:



1. The User Agent sends its request, either CDN request or HTTP request etc., to CDN A. A Request Routing System of CDN A processes the request, and it recognizes that the request is best served by another CDN, specifically CDN B.

2. A Request Routing System of CDN A sends an HTTP POST to CDN B's CDNI Request Routing Interface URI encapsulating User Agent's request, as well as a link to the associated CDNI metadata, etc., which may help CDN B make its decision when processing the request.

3. A Request Routing System of CDN B processes the request and replies an HTTP "200" Response with body containing encapsulation of protocol specific response (e.g., for DNS, CNAME of the dCDN's Request Router, and for HTTP, IP address or hostname of dCDN's Request Router) as well as parameters that indicate the properties of the response (e.g., parameter that indicate whether the response is cacheable) to return to the User Agent.

4. A Request Routing System of CDN B sends the protocol specific response to the User Agent, and then the User Agent's request will be redirect to the CDN B.

6. Protocol Specification

This section only specifies a brief introduction of different fields to be exchanged in Request Routing requests/responses, and the specific encoding for them will be presented in our future work.

6.1. Recursive Request Routing

6.1.1. DNS based Request Routing Protocol

6.1.1.1. Upstream CDN Behavior

Upon receiving a DNS query request from a User Agent, the Request Routing System of the Upstream CDN SHALL first determine an inside routing mechanism according to local policy. If it is aware that the End User is best served by another CDN, the Upstream CDN SHALL select a Downstream CDN and forward the End User's request along with a link to the associated CDNI Metadata to the Downstream CDN, while the End User's request includes DNS Client IP address, type of query, real Client IP address (if know), and URI of requested content.

Upon receiving a response from the Downstream CDN, the Request Routing System of the Upstream CDN shall forward it back to the User Agent.

6.1.1.2. Downstream CDN Behavior

Upon receiving a DNS query request, the Downstream CDN SHALL extracts End User's request information (e.g., the content provider's domain name) as well as associated CDNI Metadata to help it process the request. It then SHALL determine an inside routing mechanism according to local routing policy.

Editor's note: The local routing policy may take into account the CP's policy if existed identified by the CP's domain name.

In case of DNS Redirection, it SHALL select a Request Router and return a response containing CNAME of the Downstream CDN's Request Router, as well as parameters/policies that indicate the properties of the response, such as, parameter that indicate whether the response is cacheable, parameter that indicate how long to reduce the number of subsequent CDNI Request Routing requests the Upstream CDN needs to make, and parameter that indicate the scope of the response

(if it is cacheable).

In the case of HTTP Redirection, it SHALL select a Request Router and return IP address or hostname of Request Router, as well as parameters/policies that indicate the properties of the response, such as, parameter that indicate whether the response is cacheable, parameter that indicate how long to reduce the number of subsequent CDNI Request Routing Requests the Upstream CDN needs to make, and parameter that indicate the scope of the response (if it is cacheable).

6.1.2. HTTP based Request Routing Protocol

6.1.2.1. Upstream CDN Behavior

Upon receiving an HTTP Request from a User Agent for specific content, based on the local routing policy, if it is determined that the user is best served by another CDN, the Request Router of the Upstream CDN SHALL select a Downstream CDN for the End User, encapsulate the User Agent's request (Client IP address, request URL) with properties that Upstream CDN can use to control the dCDN's response (a link to the associated CDNI Metadata, whether to return hostnames or IP address in the redirection URL), and then forward the request to the selected Downstream CDN.

After receiving an HTTP "200" response from the Downstream CDN, the Upstream CDN SHALL forward it back to the User Agent.

6.1.2.2. Downstream CDN Behavior

Upon receiving an HTTP Request, the Downstream CDN SHALL select a delivery node for the User Agent based on the local routing policy. It SHALL then return IP address or hostname of selected delivery node, as well as parameters/policies that indicate the properties of the response, such as, parameter that indicate whether the response is cacheable, parameter that indicate how long to reduce the number of subsequent CDNI Request Routing Requests the Upstream CDN needs to make, and parameter that indicate the scope of the response (if it is cacheable).

6.2. Iterative Request Routing

Editor's note: Whether any content relative to Iterative Request Routing should be added here is to be determined by the CDNI working group.

7. Security Considerations

In HTTP based Recursive Request Routing, the end user's web browsers will not send cookies if the content request is redirected to a URL in a different domain rather than the original CP's domain, e.g. the Downstream CDN's domain. If the browser is expected to send any cookies associated with the original CP's domain, this will cause problem that the CP's policy is not enforced by the CDN.

The section 5.2 of draft [I-D.draft-peterson-cdni-strawman] has discussed a similar question and given a solution.

8. IANA Considerations

This document makes no request of IANA.

9. References

9.1. Normative Reference

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", June 1999.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax and Semantics", January 2005.

9.2. Informative Reference

- [draft-cdni-use-cases]
Bertrand, G., Emile, S., Watson, G., Burbridge, T., Eardley, P., and K. Ma, "Use Cases for Content Delivery Network Interconnection", Sep. 2011.
- [draft-cdni-problem-statement]
Niven-Jenkins, B., Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", Sep. 2011.
- [draft-cdni-requirements]
Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", Sep. 2011.

[draft-peterson-cdni-strawman]

Peterson, L. and J. Hartman, "A Simple Approach to CDN Interconnection", May 2011.

[davie-cdni-framework]

Davie, B. and L. Peterson, "A Simple Approach to CDN Interconnection", July 2011.

Authors' Addresses

Wang Danhua (editor)
Huawei Technologies
No. 101 Software Avenue
Nanjing, Jiangsu Province 210001
P.R.China

Phone: +86-25-56624734
Fax: +86-25-56624702
Email: wangdanhua@huawei.com

He Xiaoyan
Huawei
B2, Huawei Industrial Base
518129
P.R.China

Email: hexiaoyan@huawei.com

Spencer Dawkins
Huawei

Email: spencer.dawkins@wondermaster.com

Ge Chen
China Telecom
109 West Zhongshan Ave, Tianhe District
Guangzhou
P.R. China

Email: cheng@gsta.com

Ni Wei
China Mobile
No.32 Xuanwumen West Street Xicheng District
Beijing 100053
P.R. China

Email: niwei@chinamobile.com

Zhang Yunfei
China Mobile

Email: zhangyunfei@chinamobile.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 02, 2013

Danhua. Wang, Ed.
Huawei Technologies
B. Niven-Jenkins, Ed.
Velocix (Alcatel-Lucent)
Xiaoyan. He
Huawei
Chen. Ge
China Telecom
Wei. Ni
China Mobile
March 31, 2013

Request Routing Redirection Interface for CDN Interconnection
draft-he-cdni-routing-request-redirection-05

Abstract

The Request Routing Interface comprises of (1) the asynchronous advertisement of footprint and capabilities by a downstream CDN that allows a upstream CDN to decide whether to redirect particular user requests to that downstream CDN; and (2) the synchronous operation of an upstream CDN requesting whether a downstream CDN is prepared to accept a user request and of a downstream CDN responding with how to actually redirect the user request. This document describes an interface for the latter part, i.e. the CDNI request routing/Redirection Interface.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 02, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Interface function and operation overview	4
4. HTTP based RESTful interface for the Redirection Interface .	5
4.1. Information passed in RI requests & responses	7
4.2. JSON encoding of RI requests & responses	9
4.3. DNS redirection	10
4.3.1. DNS Redirection requests	10
4.3.2. DNS Redirection responses	12
4.4. HTTP Redirection	13
4.4.1. HTTP Redirection requests	13
4.4.2. HTTP Redirection responses	14
4.5. Indicating the cacheability and scope of responses . . .	15
4.6. Error responses	17
4.7. Loop detection & prevention	18
5. Security Considerations	19
6. IANA Considerations	19
7. Acknowledgements	20
8. Outstanding considerations	20
9. Contributing Authors	20
10. References	20
10.1. Normative References	20
10.2. Informative References	21
Authors' Addresses	21

1. Introduction

A Content Delivery Network (CDN) is a system built on an existing IP network which is used for large scale content delivery, via prefetching or dynamically caching content on its distributed surrogates (caching servers). [RFC6707] describes the problem area of interconnecting CDNs.

The CDNI request routing interface outlined in [I-D.ietf-cdni-framework] comprises of:

1. The asynchronous advertisement of footprint and capabilities by a downstream CDN that allows a upstream CDN to decide whether to redirect particular user requests to that downstream CDN.
2. The synchronous operation of an upstream CDN requesting whether a downstream CDN is prepared to accept a user request and of a downstream CDN responding with how to actually redirect the user request.

This document describes an interface for the latter part, i.e. the CDNI request routing/Redirection Interface (RI).

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This document reuses the terminology defined in [RFC6707]. The term "Distinguished CDN Domain" defined in [I-D.ietf-cdni-framework] is also reused in this document.

The following additional terms are introduced by this document:

Application Level Redirection: The act of using an application specific redirection mechanism for the request routing process of a CDN. The Redirection Target (RT) is the result of the routing decision of a CDN at the time it receives a content request via an application specific protocol response. Examples of an application level redirection are HTTP 302 Redirection and RTMP 302 Redirection.

DNS Redirection: The act of using DNS name resolution for the request routing process of a CDN. In DNS Redirection, the DNS name server of the CDN makes the routing decision based on a local policy and selects one or more Redirection Targets (RTs) and redirects the user agent to the RT(s) by returning the details of the RT(s) in response to the DNS query request from the user agent's DNS resolver.

HTTP Redirection: The act of using an HTTP redirection response for the request routing process of a CDN. The Redirection Target (RT) is the result of the routing decision of a CDN at the time it receives a content request via HTTP. HTTP Redirection is a particular case of Application Level Redirection.

Redirection Target (RT): A Redirection Target is the endpoint to which the user agent is redirected. In CDNI, a RT may point to a number of different components, some examples include a surrogate in the same CDN as the request router, a request router in a downstream CDN or a surrogate in a downstream CDN, etc.

3. Interface function and operation overview

[[Editor's note: Need to factor token authorisation into a future draft when that work is more stable/mature within the WG.]]

The CDNI request routing/Redirection Interface (RI) is one of the main building blocks required in order to interconnect CDNs. The main function of the Redirection Interface is to allow the Request Routing systems in interconnected CDNs to communicate to facilitate the redirection of User Agent requests between interconnected CDNs.

The detailed requirements for the Redirection Interface and their relative priorities are described in section 5 of [I-D.ietf-cdni-requirements].

The User Agent will make a request to a request router in the uCDN using one of either DNS or HTTP. If the RI is used between the uCDN and one or more dCDNs. The dCDN's RI response may contain a Redirection Target with a type that is compatible with the protocol used between User Agent and uCDN request router. The dCDN has control over the Redirection Target it provides and depending on the returned Redirection Target, the User Agent's request may be redirected to:

- o The final Surrogate, which may be in the dCDN or another dCDN (if dCDN delegates the delivery to another CDN).
- o A request router (in dCDN or another CDN) that will be using a redirection protocol (DNS or HTTP) which may or may not be the same as original redirection protocol.

The Redirection Interface operates between the Request Routing systems of a pair of interconnected CDNs. To enable communication over the Redirection Interface, the two interconnected CDNs need to know the end point (URI) in the other CDN to query. For example, an Upstream CDN needs to know the URI (end point) in a Downstream CDN to send its CDNI request routing queries to.

The Redirection Interface URI may be statically pre-configured, dynamically discovered via the CDNI control interface, or discovered via other means. However, such discovery mechanisms are not specified in this document, as they are considered out of the scope of the Redirection Interface specification.

CDNI solutions must support both of the request routing mechanisms illustrated in section 2.1 of [I-D.ietf-cdni-framework], namely Iterative Request Redirection and Recursive Request Redirection. However, the Iterative Request Redirection method does not invoke any interaction over the Redirection Interface between interconnected CDNs. Therefore, the Redirection Interface is only relevant in the case of Recursive Request Redirection and so this document will not discuss Iterative Request Redirection further.

In the case of Recursive Request Redirection, in order to perform redirection of a request received from a User Agent, the Upstream CDN queries the Downstream CDN so that the Downstream CDN can select and provide a Redirection Target. In cases where a uCDN has a choice of dCDNs it is down to the uCDN to decide (for example via configured policies) which dCDN(s) to query and in which order to query them. A number of strategies are possible including selecting a preferred dCDN based on local policy, possibly falling back to querying an alternative dCDN(s) if the first dCDN does not return a Redirection Target or otherwise reject the uCDN's RI request. A more complex strategy could be to query multiple dCDNs in parallel before selecting one and using the Redirection Target provided by that dCDN.

The Upstream CDN->User Agent redirection protocols addressed in this draft are: DNS redirection and HTTP redirection. Other types of application level redirection will not be discussed further in this draft. However the Redirection Interface is designed to be extensible and could be extended to support additional application level redirection protocols.

Also, according to the CDNI generic and request routing interface requirements, the CDNI solution shall support mechanisms to prevent and detect RI request loops. To meet such requirements, this document defines a loop prevention and detection mechanism as part of the Redirection Interface.

4. HTTP based RESTful interface for the Redirection Interface

This document defines a simple RESTful interface for the Redirection Interface based on HTTP [RFC2616], where the attributes of a User Agent's requests are encapsulated along with any other data that can aid the downstream CDN in processing the requests. The RI response encapsulates the attributes of the RT(s) that the upstream CDN should

return to the User Agent (if it decides to utilize the Downstream CDN for delivery) along with the policy for how the response can be reused.

The same RESTful interface is used for both DNS and HTTP redirection of User Agent's requests, although the contents of the RI requests/responses contain data specific to either DNS or HTTP redirection.

This approach has been chosen because it enables CDN operators to only have to deploy a single (RESTful) interface for the RI between their CDNs, regardless of the User Agent redirection method. In this way, from an operational point of view there is only one interface to monitor, manage, develop troubleshooting tools for, etc.

In addition, having a single RI where the attributes of the User Agent's DNS or HTTP request are encapsulated along with the other data required for the downstream CDN to make a request routing decision, avoids having to try and encapsulate or proxy DNS/HTTP/RTMP/etc requests and find ways to somehow embed the additional CDNI request routing/Redirection Interface properties/data within those End User DNS/HTTP/RTMP/etc requests.

Finally, the RI is easily extendable to support other User Agent request redirection methods (e.g. RTMP 302 redirection).

The generic Recursive Request Redirection message flow between Request Routing systems in a pair of interconnected CDNs is as follows:

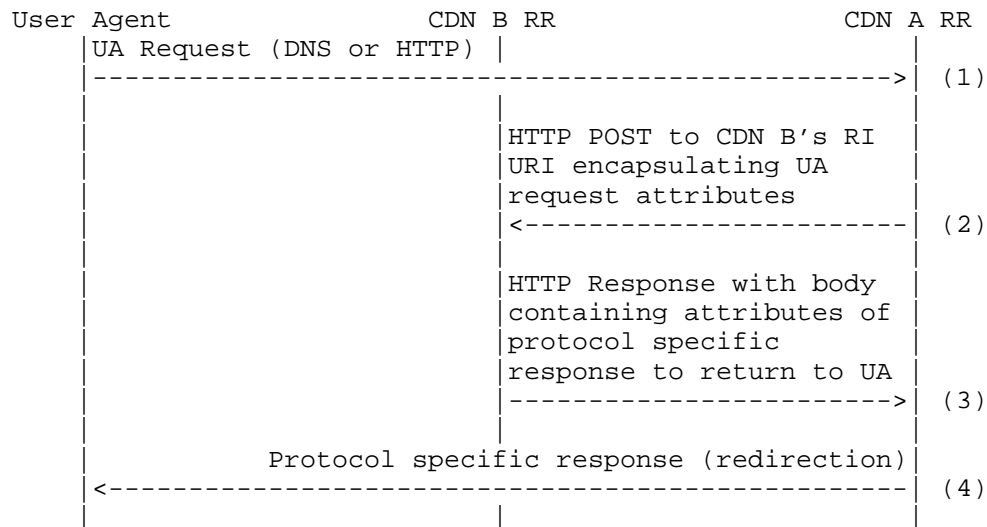


Figure 1: Generic Recursive Request Redirection message flow

1. The User Agent sends its request, either DNS request or HTTP request, to CDN A. The Request Routing System of CDN A processes the request and, through local policy, it recognizes that the request is best served by another CDN, specifically CDN B (or that CDN B is one of a number of candidate dCDNs it could use).
2. The Request Routing System of CDN A sends an HTTP POST to CDN B's RI URI containing the attributes of the User Agent's request.
3. The Request Routing System of CDN B processes the request and assuming the request is well formed, etc. responds with an HTTP "200" response with a message body containing the RT(s) to return to the User Agent as well as parameters that indicate the properties of the response (cacheability and scope).
4. The Request Routing System of CDN A sends a protocol specific response (containing the returned attributes) to the User Agent, so that the User Agent's request will be redirected to the RT(s) returned by CDN B.

4.1. Information passed in RI requests & responses

The information passed in RI requests splits into two basic categories:

1. The attributes of the User Agent's request to the upstream CDN.

2. Properties/parameters that the uCDN can use to control the dCDN's response or that can help the dCDN make its decision.

To assist the routing decision of a Downstream CDN, the Upstream CDN shall convey as much information as possible to the Downstream CDN, for example the URI of the requested content and the User Agent's location information, when those are known by the uCDN Request Routing system.

In order for the Downstream CDN to determine whether it is capable of delivering any requested content, it requires CDNI metadata related to the content the User Agent is requesting. That metadata will describe the content and any policies associated with it. It is expected that the RI request contains sufficient information for the Request Router in the Downstream CDN to be able to retrieve the require CDNI Metadata via the CDNI Metadata interface.

The information passed in RI responses splits into two basic categories:

1. The attributes of the RT to return to the User Agent in the DNS response or HTTP response.
2. Parameters/policies that indicate the properties of the response, such as, whether it is cacheable, the scope of the response, etc.

In addition to details of how to redirect the User Agent, the Downstream CDN may wish to return additional policy to the Upstream CDN to help the Upstream CDN with future RI requests. For example the Downstream CDN may wish to return a policy that expresses "this response can be reused without requiring a RI request for 60 seconds provided the User Agent's IP address is in the range 192.0.2.0 - 192.0.2.255".

These additional policies split into two basic categories:

- o An indication of the cacheability of the response carried in the HTTP response headers (to reduce the number of subsequent RI requests the uCDN needs to make).
- o The scope of the response (if it is cacheable) carried within the body of the HTTP response. For example whether the response applies to a wider range of IP addresses than what was included in the RI request.

The cacheability of the response is indicated using the standard HTTP Cache-Control mechanisms.

4.2. JSON encoding of RI requests & responses

The body of RI requests and responses is a JSON object containing a dictionary of keys. Keys MUST always be encoded in lowercase. Unknown keys MUST be ignored but the response MUST NOT be considered invalid unless the syntax of the request is invalid.

The following keys are defined:

Key	Request/Response	Description
dns	Both	The attributes of the UA's DNS request or the attributes of the RT(s) to return in a DNS response.
http	Both	The attributes of the UA's HTTP request or the attributes of the RT to return in a HTTP response.
scope	Response	The scope of the response (if it is cacheable). For example whether the response applies to a wider range of IP addresses than what was included in the RI request.
error	Response	Additional details if the response is an error response.
cdn-path	Both	A List of Strings. Contains the CDN Provider IDs of previous CDNs this RI request has passed through. When cascading a RI request the transit CDN appends its own CDN Provider ID to the list in cdn-path so that downstream CDNs can detect loops in the RI request chain. Transit CDNs should check the cdn-path and not cascade the RI request to downstream CDNs that are already listed in cdn-path. The cdn-path MUST be reflected back in RI responses.
max-hops	Request	Integer specifying the Maximum Number of hops (CDN Provider IDs) this request is allowed to be propagated along. This allows the uCDN to crudely constrain the latency of the request routing

```

|               |               | chain.               |
+-----+-----+-----+-----+

```

Top-Level keys in RI requests/responses

A single request or response MUST contain only one of the dns or http keys. Requests MUST contain a cdn-path key.

[[Editor's note: Need some text/section specifying the Media Types for RI requests/responses]]

[[Editor's note: Need some text on minimum attributes to be able to (at least parse) - e.g. A/AAAA/CNAME, etc]]

[[Editor's note: Need section detailing format/etc for scope and error keys]]

Note: All implementations MUST support IPv4 addresses encoded as specified by the 'IPv4address' rule in Section 3.2.2 of [RFC3986] and MUST support all IPv6 address formats specified in [RFC4291]. Server implementations SHOULD use IPv6 address formats specified in [RFC5952].

4.3. DNS redirection

The following sections provide more detailed descriptions of the information that should be passed in RI requests and responses for DNS redirection.

4.3.1. DNS Redirection requests

For DNS based redirection the uCDN needs to pass the following information to the dCDN in the RI request:

- o The IP address of the DNS resolver that made the DNS request to the Upstream CDN.
- o The type of DNS query made (A, AAAA, RCODEs, etc.).
- o The class of DNS query made (usually IN). [[Editor's Note: Do we need to include class or can we always assume it is IN?]]
- o The fully qualified domain name for which DNS redirection is being requested.
- o The IP address or prefix of the User Agent (if known to the Upstream CDN, e.g. through draft-vandergaast-edns-client-subnet).

The information above is encoded as a set of key:value pairs within the dns dictionary as follows:

Key	Value	Mandatory	Description
resolver-ip	String	Yes	The IP address of the UA's DNS resolver.
qtype	String	Yes	The type of DNS query made by the UA's DNS resolvers in uppercase (A, AAAA, etc.).
qclass	String	Yes	The class of DNS query made in uppercase (IN, etc.).
qname	String	Yes	The fully qualified domain name being queried.
c-subnet	String	No	The IP address of the UA in CIDR format.
dns-only	Boolean	No	If True then dCDN MUST only use DNS redirection to a surrogate and MUST include the dns-only property set to True on any cascaded RI requests. Defaults to False.

An example RI request (uCDN->dCDN) for DNS based redirection:

```
POST /dcdn/ri HTTP/1.1
Host: rr1.dcdn.example.net
Accept: application/vnd.cdni.ri.response+json
```

```
{
  "dns" : {
    "resolver-ip" : "192.0.2.1",
    "c-subnet" : "198.51.100.0/24",
    "qtype" : "A",
    "qclass" : "IN",
    "qname" : "www.example.com"
  },
  "cdn-path": ["AS65551:0"],
  "max-hops": 3
}
```

4.3.2. DNS Redirection responses

For DNS based redirection the dCDN needs to return one of the following to the uCDN in the RI response:

- o The IP address of (or a CNAME to) the RT (if the dCDN is performing DNS based redirection); or
- o The IP address of (or a CNAME to) a RT which is a Request Router (if the dCDN is performing HTTP based redirection).

The information above is encoded as a set of key:value pairs within the dns dictionary as follows:

Key	Value	Mandatory	Description
rcode	Integer	Yes	DNS response code.
name	String	Yes	The fully qualified domain name the response relates to.
a	List of String	No	Set of IPv4 Addresses of RT(s).
aaaa	List of String	No	Set of IPv6 Addresses of RT(s).
cname	List of String	No	Set of fully qualified domain names of RT(s).
ttd	Integer	No	TTL of DNS response. Default is 0.

Response must contain at least one of a, aaaa, cname.

An example of a successful RI response (dCDN->uCDN) for DNS based redirection:

[[Editor's note: Currently shows both A/AAAA & CNAME in single response, need to split to show the different use cases]]

HTTP/1.1 200 OK

Date: Mon, 06 Aug 2012 18:41:38 GMT

Content-Type: application/vnd.cdni.ri.response+json

```
{
  "dns" : {
    "rcode" : 0,
    "name" : "www.example.com",
```

```

    "a" : ["192.0.2.200", "192.0.2.201"],
    "aaaa" : ["2001:DB8::C8", "2001:DB8::C9"],
    "cname" : ["rr1.dcdn.example",
               "rr2.dcdn.example"],
    "ttl" : 60
  }
}

```

4.4. HTTP Redirection

The following sections provide more detailed descriptions of the information that should be passed in RI requests and responses for HTTP redirection.

4.4.1. HTTP Redirection requests

For HTTP based redirection the uCDN MUST pass the following information to the dCDN in the RI request:

- o The IP address of the User Agent.
- o The URL requested by the User Agent.

The uCDN MAY also pass additional information to the dCDN in the RI request, such as:

- o The HTTP method or version number of the User Agent's request.
- o Additional HTTP header included in the User Agent request.

The information above is encoded as a set of key:value pairs within the http dictionary as follows:

Key	Value	Mandatory	Description
c-ip	String	Yes	The IP address of the UA/client
cs-uri	String	Yes	The URI requested by the UA/client.
cs(<HeaderName>)	String	No	The contents of the HTTP header named <HeaderName> as a string, for example cs(Cookie) would contain the content of the HTTP Cookie: header. Two

			special <HeaderName>s are defined: cs(Method) and cs(HTTP-Version) which contain the contents of the Method & HTTP-Version parts of the Request-Line as defined in Section 5.1 of [RFC2616].
--	--	--	--

An example RI request (uCDN->dCDN) for HTTP based redirection:

```
POST/dcdn/rrri HTTP/1.1
Host: rr1.dcdn.example.net
Accept: application/vnd.cdni.rrri.response+json
```

```
{
  "http": {
    "c-ip": "198.51.100.1",
    "cs-uri": "http://www.example.com"
  },
  "cdn-path": ["AS65551:0"],
  "max-hops": 3
}
```

4.4.2. HTTP Redirection responses

For HTTP based redirection the dCDN needs to return one of the following to the uCDN in the RI response:

- o A URL pointing to the selected RT (if the dCDN is redirecting the User Agent directly to a surrogate); or
- o A URL pointing to a RT which is a Request Router (if the dCDN is not redirecting the User Agent directly to a surrogate).

The information above is encoded as a set of key:value pairs within the http dictionary as follows:

Key	Value	Mandatory	Description
sc-status	Integer	Yes	The status code of the HTTP response to return to the UA

cs-uri	String	Yes	(usually 302). The URI requested by the UA/client.
sc-location	String	Yes	The contents of the Location header to return to the UA (i.e. a URI pointing to the RT(s)).
sc-cache-control	String	No	The contents of the Cache-Control header to return to the UA.

[[Editor's Note: Should we change the format above to align with the cs() format for headers on the RI request and allow the dCDN to signal back any headers it wants in the response as sc(<HeaderName>)? How to handle sc-status in that case - as a "special" header or separate key? Probably need to give some advice on HTTP headers the uCDN may want to override/not pass through, e.g. Server:?]]

An example of a successful RI response (dCDN->uCDN) for HTTP based redirection:

```
HTTP/1.1 200 OK
Date: Mon, 06 Aug 2012 18:41:38 GMT
Content-Type: application/vnd.cdni.r1.response+json
```

```
{
  "http": {
    "sc-status": 302,
    "cs-uri": "http://www.example.com"
    "sc-location":
      "http://surl.dcdn.example/ucdn/example.com",
    "sc-cache-control" : "public, max-age=30"
  }
}
```

4.5. Indicating the cacheability and scope of responses

[[Editor's note: Need to expand text a little.]]

Cacheability is via the standard HTTP Cache-Control mechanisms.

Scope is encoded as a set of key:value pairs within the scope dictionary as follows:

Key	Value	Mandatory	Description
iprange	List of String	No	A List of IP subnets in CIDR notation that this RI response can be reused for, provided the RI response is still considered fresh.

If a uCDN has multiple cached responses with overlapping scopes, longest prefix matching of the User Agent's IP against the IP subnets in the scope of each response SHOULD be used to select the most appropriate RI response to use. [[Editor's note: is this always true? What about the most recent response, should that override older ones for the overlappign scope?]]

Example of DNS redirection response from Section 4.3.2 that is cacheable by the uCDN for 60 seconds and can be returned to any User Agent with an IPv4 address in 198.51.100.0/16.

```
HTTP/1.1 200 OK
Date: Mon, 06 Aug 2012 18:41:38 GMT
Content-Type: application/vnd.cdni.ri.response+json
Cache-Control: public, max-age=60
```

```
{
  "dns" : {
    "rcode" : 0,
    "name" : "www.example.com",
    "a" : ["192.0.2.200", "192.0.2.201"],
    "aaaa" : ["2001:DB8::C8", "2001:DB8::C9"],
    "cname" : ["rr1.dcdn.example",
               "rr2.dcdn.example"],
    "ttl" : 60
  }
  "scope" : {
    "iprange" : ["198.51.100.0/16"]
  }
}
```

Example of HTTP redirection response from Section 4.4.2 that is cacheable by the uCDN for 60 seconds and can be returned to any User Agent with an IPv4 address in 198.51.100.0/16.

```
HTTP/1.1 200 OK
Date: Mon, 06 Aug 2012 18:41:38 GMT
Content-Type: application/vnd.cdni.r1.response+json
Cache-Control: public, max-age=60
```

```
{
  "http": {
    "sc-status": 302,
    "cs-uri": "http://www.example.com"
    "sc-location":
      "http://surl.dcdn.example/ucdn/example.com",
    "sc-cache-control" : "public, max-age=30"
  }
  "scope" : {
    "iprange" : ["198.51.100.0/16"]
  }
}
```

4.6. Error responses

[[Editor's note: Probably need more explanation & examples of errors that shouldn't be propagated to the User Agent?]]

RI error response examples.

RI error response (dCDN->uCDN) for DNS based User Agent requests:

HTTP/1.1 500 Server Error
 Date: Mon, 06 Aug 2012 18:41:38 GMT
 Content-Type: application/vnd.cdni.rrri.error+json
 Cache-Control: private, no-cache

```
{
  "dns" : {
    "rcode" : 4                                # DNS response code (e.g.
                                              # doesn't support AAAA)
    "name" : "www.example.com",               # domain name response
                                              # relates to
  },
  "error" : {
    "code" : TBD,                             # Give each error type its
                                              # own numeric code
    "description" :                           # Give more informative
    "IPv6/AAAA queries are not supported"     # description than just
  }                                           # protocol specific error
                                              # codes
}
```

RI error response (dCDN->uCDN) for HTTP based User Agent requests:

HTTP/1.1 500 Server Error
 Date: Mon, 06 Aug 2012 18:41:38 GMT
 Content-Type: application/vnd.cdni.rrri.error+json
 Cache-Control: private, no-cache

```
{
  "http": {
    "rcode": 400,                             # HTTP response code
    "url": "http://www.example.com",         # URL response
                                              # relates to
  }
  "error" : {
    "code" : TBD,                             # Give each error type its
                                              # own numeric code
    "description" : TBD                       # Give more informative
                                              # description than just
  }                                           # protocol specific error
                                              # codes
}
```

4.7. Loop detection & prevention

In order to prevent and detect RI request loops, each CDN MUST insert its CDN Provider ID into the cdn-path key of every RI request it originates or cascades. When receiving RI requests a dCDN should check the cdn-path and reject any RI requests which already contain the downstream CDN's Provider ID in the cdn-path. Transit CDNs should check the cdn-path and not cascade the RI request to downstream CDNs that are already listed in cdn-path. CDNs MUST NOT propagate to any downstream CDNs if the number of CDN Provider IDs in cdn-path (including the CDN's own Provider ID) is equal to or greater than max-hops.

The CDN Provider ID uniquely identifies each CDN provider during the course of request routing redirection. It consists of the the characters AS followed by the CDN Provider's AS number, then a colon (':') and an additional qualifier that is used to guarantee uniqueness in case a particular AS has multiple independent CDNs deployed. For example "AS65551:0".

If a downstream CDN receives a RI request whose cdn-path already contains that downstream CDN's Provider ID the downstream CDN MUST send a RI response with an error code of [[TBD]].

It should be noted that the loop detection & prevention mechanisms described above only cover preventing and detecting loops within the RI itself. In the cases where the IP address(es) or URI(s) returned in RI responses do not resolve directly to a surrogate in the final dCDN it is also possible to have redirection loops where Request Routers in different CDNs direct User Agents in a loop.

5. Security Considerations

[[Editor's note: Not sure if this current text is really security considerations or whether it is better placed elsewhere in the document.]]

In HTTP based Recursive Request Redirection, the end user's web browsers will not send cookies if the content request is redirected to a URL in a different domain rather than the original CP's domain, e.g. the Downstream CDN's domain. If the browser is expected to send any cookies associated with the original CP's domain, this will cause problem that the CP's policy is not enforced by the CDN.

The section 5.2 of draft [I-D.peterson-cdni-strawman] has discussed a similar question and given a solution.

6. IANA Considerations

This document makes no request of IANA.

7. Acknowledgements

The authors would like to thank Ray Brandenburg, Taesang Choi, Francois le Faucheur and Scott Wainner for their valuable comments and input to this document.

8. Outstanding considerations

Along with the various Editor's notes in the document, the following items still need to be addressed:

- o What extra properties/fields are required to cover all DNS/HTTP redirection cases?
- o Do we need Queries other than A/AAAA & response other than A/AAAA/CNAME?
- o Response scopes other than IP address? (AS? URL match?)
- o Better Security Considerations section.
- o Description/specification for how to extend the protocol with additional optional parameters/attributes.

9. Contributing Authors

[RFC Editor Note: Please move the contents of this section to the Authors' Addresses section prior to publication as an RFC.]

Spencer Dawkins
Huawei

Email: spencer@wonderhamster.org

Yunfei Zhang

Email: hishigh@gmail.com

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

10.2. Informative References

- [RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", RFC 6707, September 2012.
- [I-D.ietf-cdni-framework]
Peterson, L. and B. Davie, "Framework for CDN Interconnection", draft-ietf-cdni-framework-03 (work in progress), February 2013.
- [I-D.ietf-cdni-requirements]
Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", draft-ietf-cdni-requirements-05 (work in progress), February 2013.
- [I-D.peterson-cdni-strawman]
Peterson, L. and J. Hartman, "A Simple Approach to CDN Interconnection", draft-peterson-cdni-strawman-01 (work in progress), May 2011.

Authors' Addresses

Wang Danhua (editor)
Huawei Technologies
No. 101 Software Avenue
Nanjing, Jiangsu Province 210001
P.R.China

Phone: +86-25-56624734
Fax: +86-25-56624702
Email: wangdanhua@huawei.com

Ben Niven-Jenkins (editor)
Velocix (Alcatel-Lucent)
3 Ely Road
Milton, Cambridge CB24 6DD
UK

Email: ben@velocix.com

He Xiaoyan
Huawei
B2, Huawei Industrial Base
518129
P.R.China

Email: hexiaoyan@huawei.com

Ge Chen
China Telecom
109 West Zhongshan Ave, Tianhe District
Guangzhou
P.R. China

Email: cheng@gsta.com

Ni Wei
China Mobile
No.32 Xuanwumen West Street Xicheng District
Beijing 100053
P.R. China

Email: niwei@chinamobile.com

Network Working Group
Internet-Draft
Obsoletes: 3466 (if approved)
Intended status: Informational
Expires: January 17, 2013

L. Peterson, Ed.
Verivue, Inc.
B. Davie
Nicira Networks, Inc.
July 16, 2012

Framework for CDN Interconnection
draft-ietf-cdni-framework-01

Abstract

This document presents a framework for Content Distribution Network Interconnection (CDNI). The purpose of the framework is to provide an overall picture of the problem space of CDNI and to describe the relationships among the various components necessary to interconnect CDNs. CDN Interconnection requires the specification of several interfaces and mechanisms to address issues such as request routing, metadata exchange, and the acquisition of content by one CDN from another. The intent of this document is to outline what each interface needs to accomplish, and to describe how these interfaces and mechanisms fit together, while leaving their detailed specification to other documents. It obsoletes RFC 3466.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 17, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
1.1. Terminology	4
1.2. Reference Model	5
1.3. Structure Of This Document	8
2. Building Blocks	9
2.1. Request Redirection	9
2.1.1. DNS Redirection	9
2.1.2. HTTP Redirection	10
3. Overview of CDNI Operation	10
3.1. Preliminaries	13
3.2. HTTP Redirect Example	14
3.2.1. Comments on the example	18
3.3. Recursive Redirection Example	19
3.3.1. Comments on the example	23
3.4. DNS-based redirection example	23
3.4.1. Comments on the example	26
3.5. Dynamic Footprint Discovery	27
3.6. Content Removal	29
3.7. Pre-Positioned Content Acquisition Example	29
3.8. Asynchronous CDNI Metadata Example	31
3.9. Synchronous CDNI Metadata Acquisition Example	33
3.10. Content Acquisition with Multiple Upstream CDNs	36
4. Main Interfaces	37
4.1. In-Band versus Out-of-Band Interfaces	37
4.2. Cross Interface Concerns	38
4.3. Request Routing Interface	38
4.4. Logging Interface	40
4.5. Control Interface	42
4.6. Metadata Interface	42
5. Deployment Models	43
5.1. Meshed CDNs	44
5.2. CSP combined with CDN	44
5.3. CSP using CDNI Request Routing Interface	45
5.4. CDN Federations and CDN Exchanges	46
6. Trust Model	49
7. IANA Considerations	50
8. Security Considerations	50
8.1. Security of CDNI Interfaces	51
8.2. Digital Rights Management	52
9. Contributors	52
10. Acknowledgements	52
11. Informative References	52
Authors' Addresses	54

1. Introduction

The interconnection of Content Distribution Networks (CDNs) is motivated by several use cases, such as those described in [I-D.ietf-cdni-use-cases]. The overall problem space for CDN Interconnection is described in [I-D.ietf-cdni-problem-statement]. The purpose of this document is to provide an overview of the various components necessary to interconnect CDNs. CDN Interconnection requires the specification of several interfaces and mechanisms to address issues such as request routing, metadata exchange, and the acquisition of content by one CDN from another. The intent of this document is to describe how these interfaces and mechanisms fit together, leaving their detailed specification to other documents. We make extensive use of message flow examples to illustrate the operation of interconnected CDNs, but these examples should be considered illustrative rather than prescriptive.

RFC 3466 uses different terminology and models for "Content Internetworking (CDI)". It is also less prescriptive in terms of interfaces. To avoid confusion, this document obsoletes RFC 3466.

1.1. Terminology

This document draws freely on the core terminology defined in [I-D.ietf-cdni-problem-statement]. It also introduces the following terms:

CDN Domain: a host name (FQDN) at the beginning of a URL, representing a set of content that is served by a given CDN. For example, in the URL `http://cdn.csp.com/...rest of url...`, the CDN domain is `cdn.csp.com`. A major role of CDN Domain is to identify a region (subset) of the URI space relative to which various CDN Interconnection rules and policies are to apply. For example, a record of CDN Metadata might be defined for the set of resources corresponding to some CDN Domain.

Distinguished CDN Domain: a CDN domain that is allocated by a CDN for the purposes of communication with a peer CDN, but which is not found in client requests. Such CDN domains may be used for inter-CDN acquisition, or as redirection targets, and enable a CDN to distinguish a request from a peer CDN from an end-user request.

Delivering CDN: the CDN that ultimately delivers a piece of content to the end-user. The last in a potential sequence of downstream CDNs.

Recursive CDNI request routing: When an Upstream CDN elects to redirect a request towards a Downstream CDN, the Upstream CDN can

query the Downstream CDN Request Routing system via the CDNI Request Routing interface (or use information cached from earlier similar queries) to find out how the Downstream CDN wants the request to be redirected, which allows the Upstream CDN to factor in the Downstream CDN response when redirecting the user agent. This approach is referred to as "recursive" CDNI request routing. Note that the Downstream CDN may elect to have the request redirected directly to a Surrogate inside the Downstream CDN, to the Request-Routing System of the Downstream CDN, to another CDN, or to any other system that the Downstream CDN sees as fit for handling the redirected request.

Iterative CDNI Request Routing: When an Upstream CDN elects to redirect a request towards a Downstream CDN, the Upstream CDN can base its redirection purely on a local decision (and without attempting to take into account how the Downstream CDN may in turn redirect the user agent). In that case, the Upstream CDN redirects the request to the request routing system in the Downstream CDN, which in turn will decide how to redirect that request: this approach is referred to as "iterative" CDNI request routing.

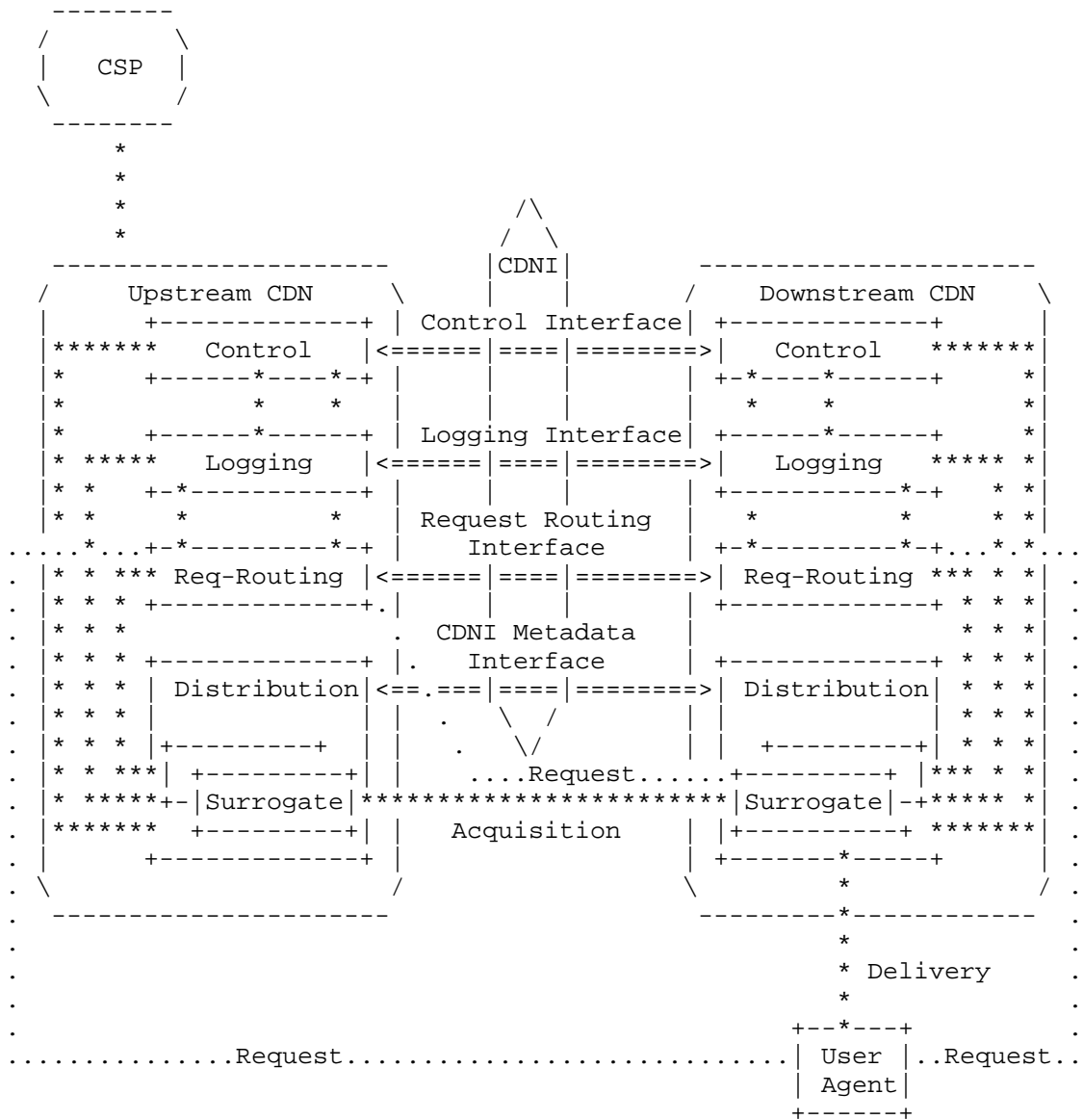
Synchronous CDNI operations: operations between CDNs that happen during the process of servicing a user request, i.e. between the time that the user agent begins its attempt to obtain content and the time at which that request is served.

Asynchronous CDNI operations: operations between CDNs that happen independently of any given user request, such as advertisement of footprint information or pre-positioning of content for later delivery.

Trigger Interface: a sub-set of the Control Interface that includes operations to pre-position, revalidate, and purge both metadata and content. These operations are typically called in response to some action (trigger) by the CSP on the upstream CDN.

1.2. Reference Model

This document uses the reference model in Figure 1 as originally created in [I-D.ietf-cdni-problem-statement].



`<==>` interfaces inside the scope of CDNI

```
**** interfaces outside the scope of CDNI
```

```
.... interfaces outside the scope of CDNI
```

Figure 1: CDNI Model and CDNI Interfaces

We note that while some interfaces in the reference model are "out of scope" for the CDNI WG (in the sense that there is no need to define new protocols for those interfaces) we still need to refer to them in this document to explain the overall operation of CDNI.

We also note that, while we generally show only one uCDN serving a given CSP, it is entirely possible that multiple uCDNs can serve a single CSP. In fact, this situation effectively exists today in the sense that a single CSP can currently connect to more than one CDN.

Definitions of the four CDNI interfaces follow. More discussion of these interfaces appears in Section 4.

- o Control Interface: Operations to discover, initialize, and parameterize the other CDNI interfaces, as well as operations to pre-position, revalidate, and purge both metadata and content. The latter sub-set of operations is sometimes collectively called the "trigger interface."
- o Request Routing Interface: Operations to determine what CDN (and optionally what surrogate within a CDN) is to serve end-user's requests. May include a combination of:
 - * Asynchronous operations to exchange routing information (e.g., the network footprint served by a given CDN) that enables CDN selection for subsequent user requests; and
 - * Synchronous operations to select a delivery CDN (surrogate) for a given user request.
- o Metadata Interface: Operations to communicate metadata that governs the how content is delivered by interconnected CDNs. Examples of CDNI metadata include geo-blocking directives, availability windows, access control mechanisms, and purge directives. May include a combination of:
 - * Asynchronous operations to exchange metadata that govern subsequent user requests for content; and
 - * Synchronous operations that govern behavior for a given user request for content.
- o Logging Interface: Operations that allow interconnected CDNs to exchange relevant activity logs. May include a combination of:
 - * Real-time exchanges, suitable for runtime traffic monitoring; and

- * Off-line exchanges, suitable for analytics and billing.

There is some ambiguity as to the line between the set of trigger-based operations in the Control interface and the Metadata interface. For both cases, the information passed from the upstream CDN to the downstream CDN can broadly be viewed as metadata that describes how content is to be managed by the downstream CDN. For example, the information conveyed by Control operations to pre-position, revalidate or purge metadata is similar to the information conveyed by posting updated metadata via the Metadata interface? Even the Control operation to purge content could be viewed as an metadata update for that content: purge simply says that the availability window for the named content ends now. The two interfaces share much in common, so minimally, there will need to be a consistent data model that spans both.

The distinction we draw has to do with what the caller knows the metadata being applied to content delivery by the callee. In the case of the Control interface, the downstream CDN returning a successful status message guarantees that the operation has been successfully completed; e.g., the content has been purged or pre-positioned. This implies that the downstream CDN accepts responsibility for having successfully completed the requested operation. In contrast, metadata passed between CDNs via the Metadata interface carries no such completion guarantee. Returning success implies successful receipt of the metadata, but nothing can be inferred about precisely when the metadata will take effect in the downstream CDN, only that it will take effect eventually. This is because of the challenge in globally synchronizing updates to metadata with end-user requests that are currently in progress (or indistinguishable from currently being in progress). Clearly, a CDN will not be viewed as a trusted peer if "eventually" often becomes an indefinite period of time, but the acceptance of responsibility cannot be as crisp for the Metadata interface.

1.3. Structure Of This Document

The remainder of this document is organized as follows:

- o Section 2 describes some essential building blocks for CDNI, notably the various options for redirecting user requests to a given CDN.
- o Section 3 provides a number of illustrative examples of various CDNI operations.
- o Section 4 describes the functionality of the four main CDNI interfaces.

- o Section 5 shows how various deployment models of CDNI may be achieved using the defined interfaces.
- o Section 6 describes the trust model of CDNI and the issues of transitive trust in particular that CDNI raises.

2. Building Blocks

2.1. Request Redirection

At its core, CDN Interconnection requires the redirection of requests from one CDN to another. For any given request that is received by an upstream CDN, it will either respond to the request directly, or somehow redirect the request to a downstream CDN. Two main mechanisms are available for redirecting a request to a downstream CDN. The first leverages the DNS name resolution process and the second uses in-protocol redirection mechanisms such as the HTTP 302 redirection response. We discuss these below as background before discussing some examples of their use in Section 3.

2.1.1. DNS Redirection

DNS redirection is based on returning different IP addresses for the same DNS name, for example, to balance server load or to account for the client's location in the network. A DNS server, sometimes called the Local DNS (LDNS), resolves DNS names on behalf of an end-user. The LDNS server in turn queries other DNS servers until it reaches the authoritative DNS server for the CDN-domain. The network operator typically provides the LDNS server, although the user is free to choose other DNS servers (e.g., OpenDNS, Google Public DNS).

The advantage of DNS redirection is that it is completely transparent to the end user--the user sends a DNS name to the LDNS server and gets back an IP address. On the other hand, DNS redirection is problematic because the DNS request comes from the LDNS server, not the end-user. This may affect the accuracy of server selection that is based on the user's location. The transparency of DNS redirection is also a problem in that there is no opportunity to modify the path component of the URL being accessed by the client. We consider two main forms of DNS redirection: simple and CNAME-based.

In simple DNS redirection, the authoritative DNS server for the name simply returns an IP address from a set of possible IP addresses. The answer is chosen from the set based on characteristics of the set (e.g., the relative loads on the servers) or characteristics of the client (e.g., the location of the client relative to the servers). Simple redirection is straightforward. The only caveats are (1)

there is a limit to the number of delivery nodes a single DNS server can manage; and (2) DNS responses are cached by downstream servers so the TTL on the response must be set to an appropriate value so as to preserve the timeliness of the redirection.

In CNAME-based DNS redirection, the authoritative server returns a CNAME response to the DNS request, telling the LDNS server to restart the name lookup using a new name. A CNAME is essentially a symbolic link in the DNS namespace, and like a symbolic link, redirection is transparent to the client--the LDNS server gets the CNAME response and re-executes the lookup. Only when the name has been resolved to an IP address does it return the result to the user. Note that DNAME would be preferable to CNAME if it becomes widely supported.

2.1.2. HTTP Redirection

HTTP redirection makes use of the "302" redirection response of the HTTP protocol. This response contains a new URL that the application should fetch instead of the original URL. By changing the URL appropriately, the server can cause the user to redirect to a different server. The advantages of 302 redirection are that (1) the server can change the URL fetched by the client to include, for example, both the DNS name of the particular server to use, as well as the original HTTP server that was being accessed; and (2) the client sends the HTTP request to the server, so that its IP address is known and can be used in selecting the server.

The disadvantages of HTTP redirection are (1) it is visible to the application, so it requires application support and may affect the application behavior (e.g., web browsers will not send cookies if the URL changes to a different domain); (2) HTTP is a heavy-weight protocol layered on TCP so it has relatively high overhead; and (3) the results of HTTP redirection are not cached so that all redirections must go through to the server.

3. Overview of CDNI Operation

To provide a big-picture overview of the various components of CDN Interconnection, we walk through a "day in the life" of a content item that is made available via a pair of interconnected CDNs. This will serve to illustrate many of the functions that need to be supported in a complete CDNI solution. We give examples using both DNS-based and HTTP-based redirection. We begin with very simple examples and then how additional capabilities, such as recursive request redirection and content removal, might be added.

Before walking through some specific examples, we present a high-

level view of the operations that may take place. This high-level overview is illustrated in Figure 2. Note that most operations will involve only a subset of all the messages shown below, and that the order and number of operations may vary considerably, as more detailed examples illustrate below.

The following shows Operator A as the upstream CDN (uCDN) and Operator B as the downstream CDN (dCDN), where the former has a relationship with a content provider and the latter being the best CDN to deliver content to the end-user. The interconnection relationship may be symmetric between these two CDN operators, but for simplicity we show the interaction in one direction only.

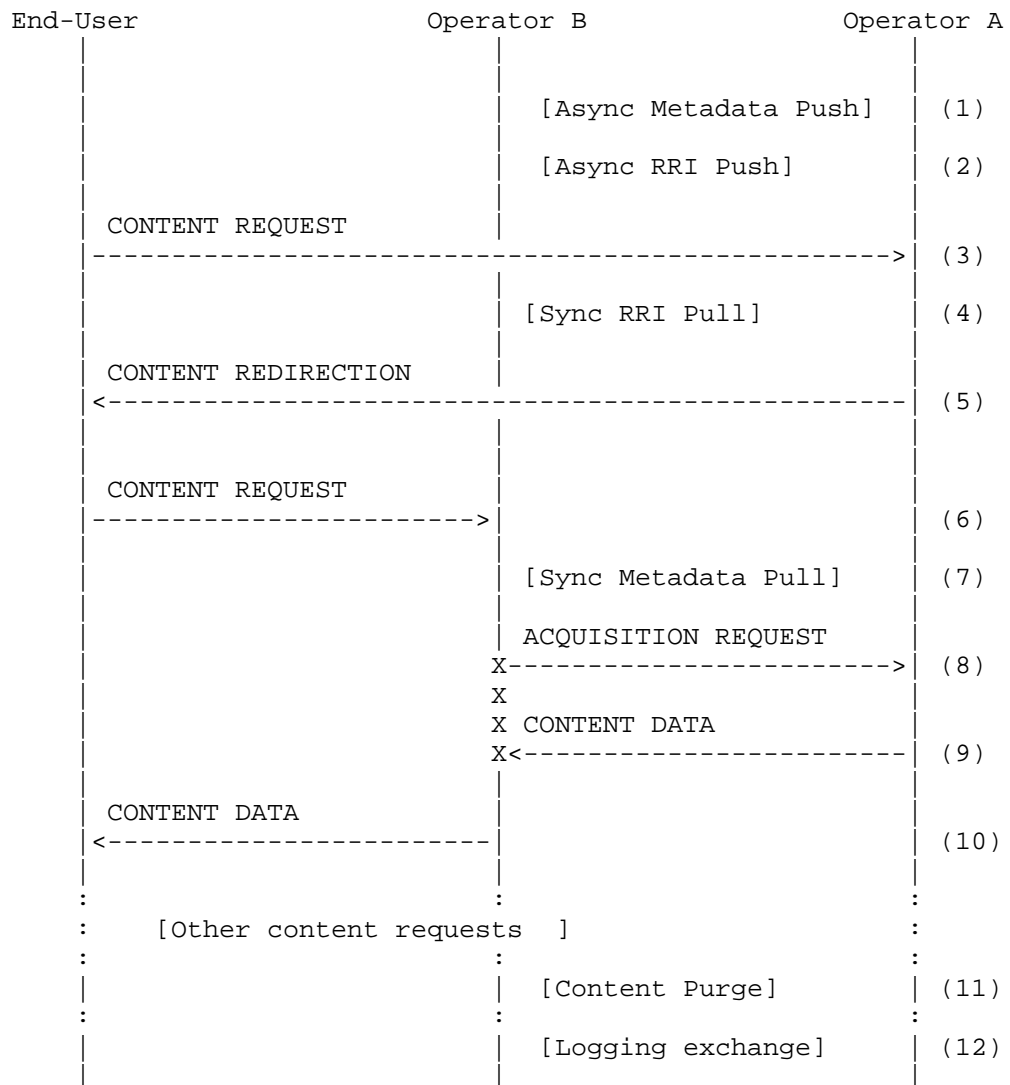


Figure 2: Overview of Operation

The operations shown in the Figure are as follows:

1. Prior to any content request, metadata may be asynchronously pushed from uCDN to dCDN so that it is available in readiness for later content requests.

2. dCDN may advertise information relevant to its delivery capabilities (e.g. geographic footprint, reachable address prefixes) prior to any content requests being redirected.
3. A content request from a user agent arrives at uCDN.
4. uCDN may synchronously request information from dCDN regarding its delivery capabilities to decide if dCDN is a suitable target for redirection of this request.
5. uCDN redirects the request to dCDN by sending some response (DNS, HTTP) to the user agent.
6. The user agent requests the content from dCDN.
7. dCDN may synchronously request metadata related to this content from uCDN, e.g. to decide whether to serve it.
8. If the content is not already in a suitable cache in dCDN, dCDN may acquire it from uCDN.
9. The content is delivered to dCDN from uCDN.
10. The content is delivered to the user agent by dCDN.
11. Some time later, perhaps at the request of the CSP (not shown) uCDN may instruct dCDN to purge the content to ensure it is not delivered again.
12. After one or more content delivery actions by dCDN, a log of delivery actions may be provided to uCDN.

The following sections show some more specific examples of how these operations may be combined to perform various delivery, control and logging operations across a pair of CDNs.

3.1. Preliminaries

Initially, we assume that there is at least one CSP that has contracted with an upstream CDN (uCDN) to deliver content on its behalf. We are not particularly concerned with the interface between the CSP and uCDN, other than to note that it is expected to be the same as in the "traditional" (non-interconnected) CDN case. Existing mechanisms such as DNS CNAMEs or HTTP redirects (Section 2) can be used to direct a user request for a piece of content from the CSP towards the CSP's chosen upstream CDN.

We use the term "CDN-domain" to refer to the host name (a FQDN) at

the beginning of each URL. We assume Operator A provides an upstream CDN that serves content on behalf of a CSP with CDN-domain `cdn.csp.com`. We assume that Operator B provides a downstream CDN. An end user at some point makes a request for URL

`http://cdn.csp.com/...rest of url...`

It may well be the case that `cdn.csp.com` is just a CNAME for some other CDN-domain (such as `csp.op-a.net`). Nevertheless, the HTTP request in the examples that follow is assumed to be for the example URL above.

Our goal is to enable content identified by the above URL to be served by the CDN of operator B. In the following sections we will walk through some scenarios in which content is served, as well as other CDNI operations such as the removal of content from a downstream CDN.

3.2. HTTP Redirect Example

In this section we walk through a simple, illustrative example using HTTP redirection from uCDN to dCDN. The example also assumes the use of HTTP redirection inside uCDN and dCDN; however, this is independent of the choice of redirection approach across CDNs, so an alternative example could be constructed still showing HTTP redirection from uCDN to dCDN but using DNS for handling of request inside each CDN.

We assume for this example that Operators A and B have established an agreement to interconnect their CDNs, with A being upstream and B being downstream. (It is likely that the agreement would be made in both directions, but we focus on just one here for clarity.)

The operators agree that a CDN-domain `peer-a.op-b.net` will be used as the target of redirections from uCDN to dCDN. The name of this domain must be communicated by some means to each CDN. (This could be established out-of-band or via a CDNI interface.) We refer to this domain as a "distinguished" CDN domain to convey the fact that its use is limited to the interconnection mechanism; such a domain is never embedded in URLs that end-users request.

The operators must also agree on some distinguished CDN-domain that will be used for inter-CDN acquisition of CSP's content from uCDN by dCDN. In this example, we'll use `op-b-acq.op-a.net`.

The operators must also exchange information regarding which requests dCDN is prepared to serve. For example, dCDN may be prepared to serve requests from clients in a given geographical region or a set

of IP address prefixes. This information may again be provided out of band or via a defined interface.

DNS must be configured in the following way:

- o The content provider must be configured to make operator A the authoritative DNS server for `cdn.csp.com` (or to return a CNAME for `cdn.csp.com` for which operator A is the authoritative DNS server).
- o Operator A must be configured so that a DNS request for `op-b-acq.op-a.net` returns a request router in Operator A.
- o Operator B must be configured so that a DNS request for `peer-a.op-b.net/cdn.csp.com` returns a request router in Operator B.

Figure 3 illustrates how a client request for

`http://cdn.csp.com/...rest of url...`

is handled.



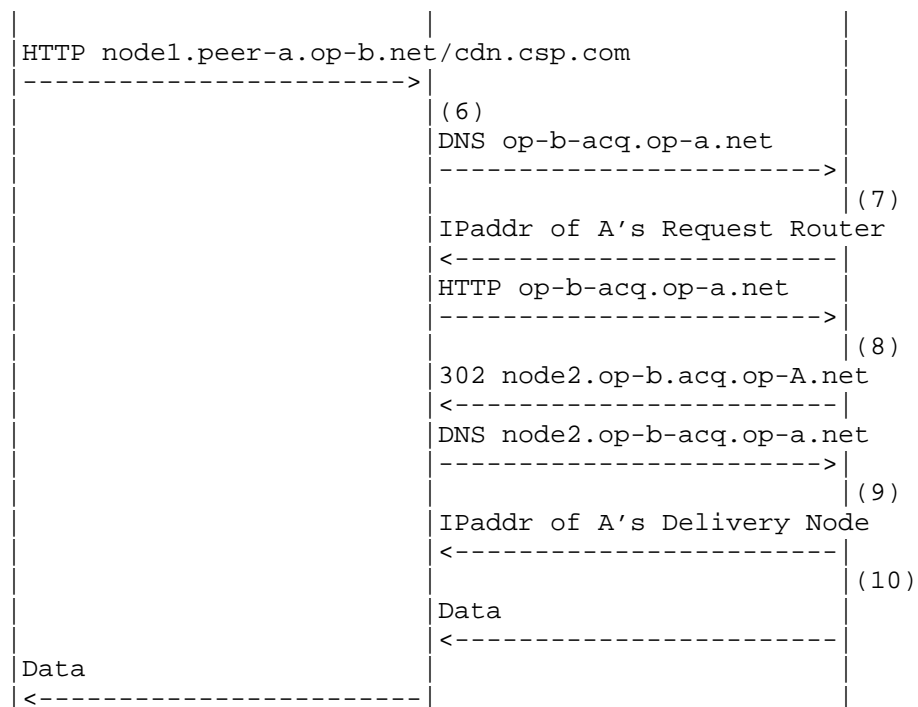


Figure 3: Request Trace for HTTP redirection method

The steps illustrated in the figure are as follows:

1. A DNS resolver for Operator A processes the DNS request for its customer based on CDN-domain `cdn.csp.com`. It returns the IP address of a request router in Operator A.
2. A Request Router for Operator A processes the HTTP request and recognizes that the end-user is best served by another CDN--specifically one provided by Operator B--and so it returns a 302 redirect message for a new URL constructed by "stacking" Operator B's distinguished CDN-domain (`peer-a.op-b.net`) on the front of the original URL. (Note that more complex URL manipulations are possible, such as replacing the initial CDN-domain by some opaque handle.)
3. The end-user does a DNS lookup using Operator B's distinguished CDN-domain (`peer-a.op-b.net`). B's DNS resolver returns the IP address of a request router for Operator B. Note that if request routing within dCDN was performed using DNS instead of HTTP redirection, B's DNS resolver would also behave as the request router and directly return the IP address of a delivery node.

4. The request router for Operator B processes the HTTP request and selects a suitable delivery node to serve the end-user request, and returns a 302 redirect message for a new URL constructed by replacing the hostname by a subdomain of the Operator B's distinguished CDN-domain that points to the selected delivery node.
5. The end-user does a DNS lookup using Operator B's delivery node subdomain (nodel.peer-a.op-b.net). B's DNS resolver returns the IP address of the delivery node.
6. The end-user requests the content from B's delivery node. In the case of a cache hit, steps 6, 7, 8, 9 and 10 below do not happen, and the content data is directly returned by the delivery node to the end-user. In the case of a cache miss, the content needs to be acquired by dCDN from uCDN (not the CSP). The distinguished CDN-domain peer-a.op-b.net indicates to dCDN that this content is to be acquired from uCDN; stripping the CDN-domain reveals the original CDN-domain cdn.csp.com and dCDN may verify that this CDN-domain belongs to a known peer (so as to avoid being tricked into serving as an open proxy). It then does a DNS request for an inter-CDN acquisition CDN-domain as agreed above (in this case, op-b-acq.op-a.net).
7. Operator A's DNS resolver processes the DNS request and returns the IP address of a request router in operator A.
8. The request router for Operator A processes the HTTP request from Operator B delivery node. Operator A request router recognizes that the request is from a peer CDN rather than an end-user because of the dedicated inter-CDN acquisition domain (op-b-acq.op-a.net). (Note that without this specially defined inter-CDN acquisition domain, operator A would be at risk of redirecting the request back to operator B, resulting in an infinite loop). The request router for Operator A selects a suitable delivery node in uCDN to serve the inter-CDN acquisition request and returns a 302 redirect message for a new URL constructed by replacing the hostname by a subdomain of the Operator A's distinguished inter-CDN acquisition domain that points to the selected delivery node.
9. Operator A DNS resolver processes the DNS request and returns the IP address of the delivery node in operator A.
10. Operator A serves content for the requested CDN-domain to dCDN. Although not shown, it is at this point that Operator A processes the rest of the URL: it extracts information identifying the origin server, validates that this server has

been registered, and determines the content provider that owns the origin server. It may also perform its own content acquisition steps if needed before returning the content to dCDN.

3.2.1. Comments on the example

The main advantage of this design is that it is simple: each CDN need only know the distinguished CDN-domain for each peer, with the upstream CDN "pushing" the downstream CDN-domain onto the URL as part of its redirect (step 2) and the downstream CDN "popping" its CDN-domain off the URL to expose a CDN-domain that the upstream CDN can correctly process. Neither CDN needs to be aware of the internal structure of the other's URLs. Moreover, the inter-CDN redirection is entirely supported by a single HTTP redirect; neither CDN needs to be aware of the other's internal redirection mechanism (i.e., whether it is DNS or HTTP based).

One disadvantage is that the end-user's browser is redirected to a new URL that is not in the same domain of the original URL. This has implications on a number of security or validation mechanisms sometimes used on endpoints. For example, it is important that any redirected URL be in the same domain (e.g., csp.com) if the browser is expected to send any cookies associated with that domain. As another example, some video players enforce validation of a cross domain policy that needs to allow for the domains involved in the CDN redirection. These problems are generally soluble, but the solutions complicate the example, so we do not discuss them further in this version of the draft.

We note that this example begins to illustrate some of the interfaces that may be required for CDNI, but does not require all of them. For example, obtaining information from dCDN regarding the set of client IP addresses or geographic regions it might be able to serve is an aspect of the request routing interface. Important configuration information such as the distinguished names used for redirection and inter-CDN acquisition could also be conveyed via a CDNI interface (e.g., perhaps the control interface). The example also shows how existing HTTP-based methods suffice for the acquisition interface. Arguably, the absolute minimum metadata required for CDNI is the information required to acquire the content, and this information was provided "in-band" in this example by means of the URI handed to the client in the HTTP 302 response. Hence, there is no explicit metadata interface invoked in this example. There is also no explicit logging interface discussed in this example.

We also note that the step of deciding when a request should be redirected to dCDN rather than served by uCDN has been somewhat

glossed over. It may be as simple as checking the client IP address against a list of prefixes, or it may be considerably more complex, involving a wide range of factors, such as the geographic location of the client (perhaps determined from a third party service), CDN load, or specific business rules.

This example uses the "iterative" CDNI request routing approach. That is, uCDN performs part of the request routing function to determine that dCDN should serve the request, and then redirects the client to a request router in dCDN to perform the rest of the request routing function. If request routing is performed in the dCDN using HTTP redirection, this translates in the end-user experiencing two successive HTTP redirections. By contrast, the alternative approach of "recursive" CDNI request routing effectively coalesces these two successive HTTP redirections into a single one, sending the end-user directly to the right delivery node in the dCDN. This "recursive" CDNI request routing approach is discussed in the next section.

3.3. Recursive Redirection Example

The following example builds on the previous one to illustrate the use of the Request Routing interface to enable "recursive" CDNI request routing. We build on the HTTP-based redirection approach because it illustrates the principles and benefits clearly, but it is equally possible to perform recursive redirection when DNS-based redirection is employed.

In contrast to the prior example, the operators need not agree in advance on a CDN-domain to serve as the target of redirections from uCDN to dCDN. The operators still must agree on some distinguished CDN-domain that will be used for inter-CDN acquisition of CSP's content by dCDN. In this example, we'll use op-b-acq.op-a.net.

The operators must also exchange information regarding which requests dCDN is prepared to serve. For example, dCDN may be prepared to serve requests from clients in a given geographical region or a set of IP address prefixes. This information may again be provided out of band or via a defined protocol.

DNS must be configured in the following way:

- o The content provider must be configured to make operator A the authoritative DNS server for cdn.csp.com (or to return a CNAME for cdn.csp.com for which operator A is the authoritative DNS server).
- o Operator A must be configured so that a DNS request for op-b-acq.op-a.net returns a request router in Operator A.

- o Operator B must be configured so that a request for `node1.opb.net/cdn.csp.com` returns the IP address of a delivery node. Note that there might be a number of such delivery nodes.

Figure 3 illustrates how a client request for

`http://cdn.csp.com/...rest of url...`

is handled.

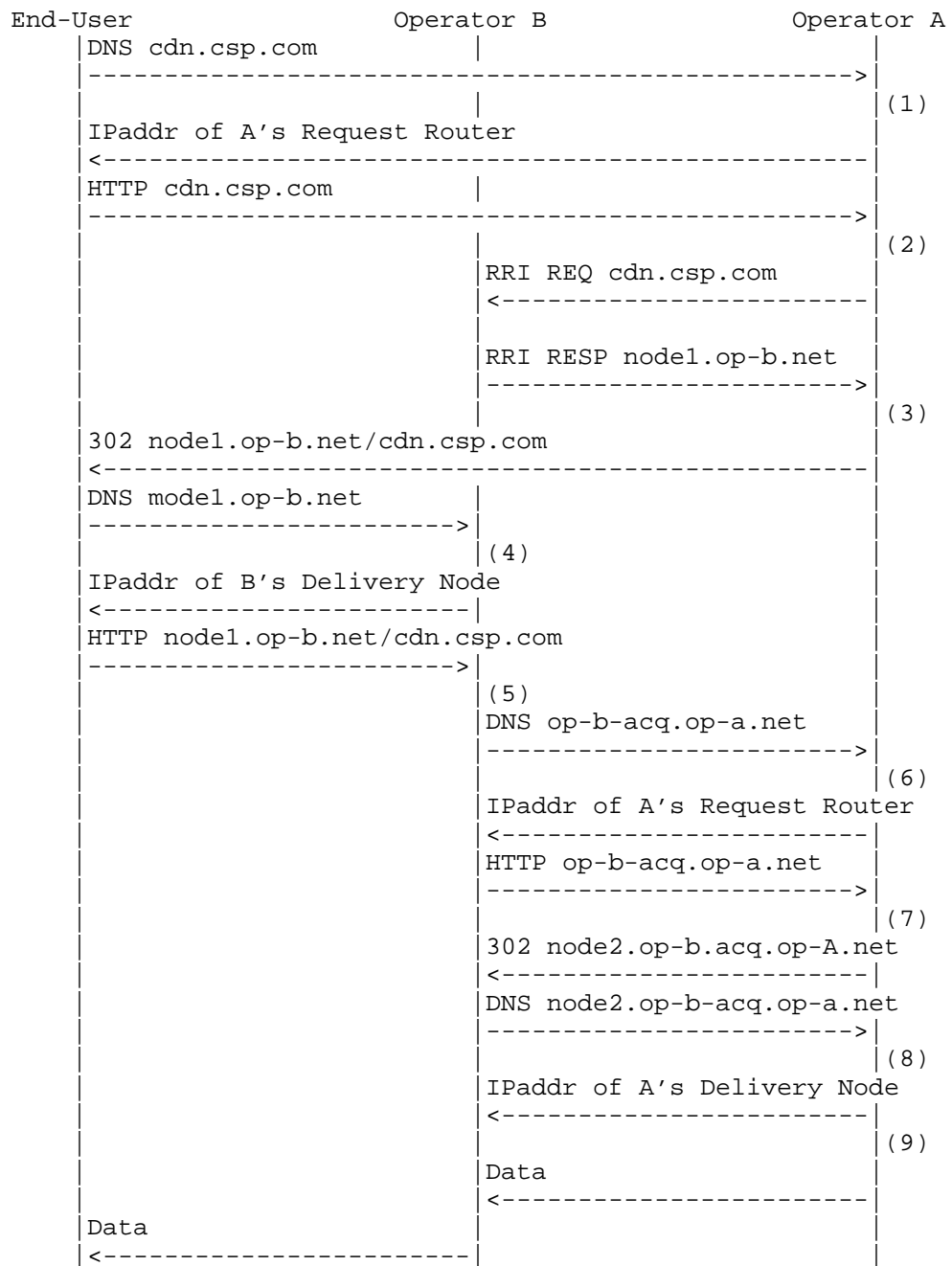


Figure 4: Request Trace for Recursive HTTP redirection method

The steps illustrated in the figure are as follows:

1. A DNS resolver for Operator A processes the DNS request for its customer based on CDN-domain `cdn.csp.com`. It returns the IP address of a Request Router in Operator A.
2. A Request Router for Operator A processes the HTTP request and recognizes that the end-user is best served by another CDN--specifically one provided by Operator B--and so it queries the CDNI Request Routing interface of Operator B, providing a set of information about the request including the URL requested. Operator B replies with the DNS name of a delivery node.
3. Operator A returns a 302 redirect message for a new URL obtained from the Request Routing Interface.
4. The end-user does a DNS lookup using the host name of the URL just provided (`node1.op-b.net`). B's DNS resolver returns the IP address of the corresponding delivery node. Note that, since the name of the delivery node was already obtained from B using the CDNI Request Routing Interface, there should not be any further redirection here (in contrast to the iterative method described above.)
5. The end-user requests the content from B's delivery node, potentially resulting in a cache miss. In the case of a cache miss, the content needs to be acquired from uCDN (not the CSP.) The distinguished CDN-domain `op-b.net` indicates to dCDN that this content is to be acquired from another CDN; stripping the CDN-domain reveals the original CDN-domain `cdn.csp.com`, dCDN may verify that this CDN-domain belongs to a known peer (so as to avoid being tricked into serving as an open proxy). It then does a DNS request for the inter-CDN Acquisition "distinguished" CDN-domain as agreed above (in this case, `op-b-acq.op-a.net`).
6. Operator A DNS resolver processes the DNS request and returns the IP address of a request router in operator A.
7. The request router for Operator A processes the HTTP request from Operator B delivery node. Operator A request router recognizes that the request is from a peer CDN rather than an end-user because of the dedicated inter-CDN acquisition domain (`op-b-acq.op-a.net`). (Note that without this specially defined inter-CDN acquisition domain, operator A would be at risk of redirecting the request back to operator B, resulting in an infinite loop). The request router for Operator A selects a suitable delivery node in uCDN to serve the inter-CDN acquisition request and returns a 302 redirect message for a new URL

constructed by replacing the hostname by a subdomain of the Operator A's distinguished inter-CDN acquisition domain that points to the selected delivery node.

8. Operator A recognizes that the DNS request is from a peer CDN rather than an end-user (due to the internal CDN-domain) and so returns the address of a delivery node. (Note that without this specially defined internal domain, Operator A would be at risk of redirecting the request back to Operator B, resulting in an infinite loop.)
9. Operator A serves content for the requested CDN-domain to dCDN. Although not shown, it is at this point that Operator A processes the rest of the URL: it extracts information identifying the origin server, validates that this server has been registered, and determines the content provider that owns the origin server. It may also perform its own content acquisition steps if needed before returning the content to dCDN.

3.3.1. Comments on the example

Recursive redirection has the advantage over iterative of being more transparent from the end-user's perspective, but the disadvantage of each CDN exposing more of its internal structure (in particular, the addresses of edge caches) to peer CDNs. By contrast, iterative redirection does not require dCDN to expose the addresses of its edge caches to uCDN.

This example happens to use HTTP-based redirection in both CDN A and CDN B, but a similar example could be constructed using DNS-based redirection in either CDN. Hence, the key point to take away here is simply that the end user only sees a single redirection of some type, as opposed to the pair of redirections in the prior (iterative) example.

The use of the Request Routing Interface requires that interface to be appropriately configured and bootstrapped, which is not shown here. More discussion on the bootstrapping of interfaces is provided in Section 4

3.4. DNS-based redirection example

In this section we walk through a simple example using DNS-based redirection for request redirection from uCDN to dCDN (as well as for request routing inside dCDN and uCDN). As noted in Section 2.1, DNS-based redirection has certain advantages over HTTP-based redirection (notably, it is transparent to the end-user) as well as some drawbacks (notably the client IP address is not visible to the

request router).

As before, Operator A must learn the set of requests that dCDN is willing or able to serve (e.g. which client IP address prefixes or geographic regions are part of the dCDN footprint). Operator B must have and make known to operator A some unique identifier that can be used for the construction of a distinguished CDN domain, as shown in more detail below. (This identifier strictly needs only to be unique within the scope of Operator A, but a globally unique identifier, such as an AS number assigned to B, is one easy way to achieve that.) Also, Operator A must obtain the NS records for Operator B's externally visible redirection servers. Also, as before, a distinguished CDN-domain, such as op-b-acq.op-a.net, must be assigned for inter-CDN acquisition.

DNS must be configured in the following way:

- o The CSP must be configured to make Operator A the authoritative DNS server for cdn.csp.com (or to return a CNAME for cdn.csp.com for which operator A is the authoritative DNS server).
- o When uCDN sees a request best served by dCDN, it returns CNAME and NS records for "b.cdn.csp.com", where "b" is the unique identifier assigned to Operator B. (It may, for example, be an AS number assigned to Operator B.)
- o dCDN must be configured so that a request for "b.cdn.csp.com" returns a delivery node in dCDN.
- o uCDN must be configured so that a request for "op-b-acq.op-a.net" returns a delivery node in uCDN.

Figure 5 depicts the exchange of DNS and HTTP requests. The main differences from Figure 3 are the lack of HTTP redirection and transparency to the end-user.

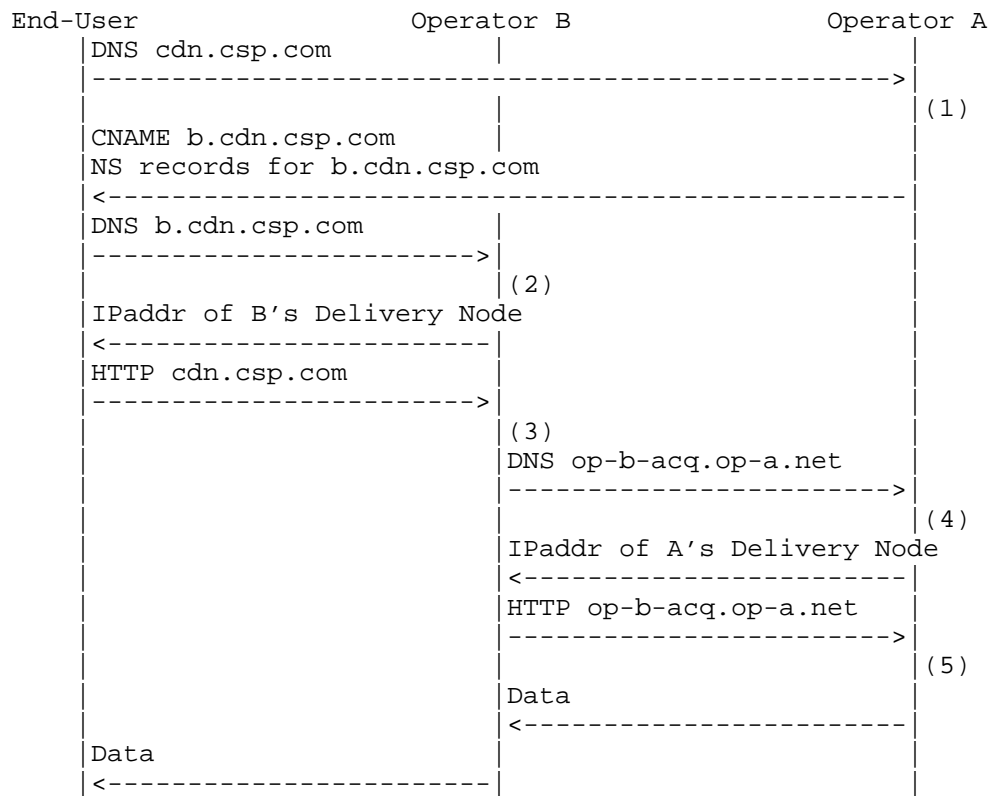


Figure 5: Request Trace for DNS-based Redirection Example

The steps illustrated in the figure are as follows:

1. Request Router for Operator A processes the DNS request for CDN-domain `cdn.csp.com` and recognizes that the end-user is best served by another CDN. (This may depend on the IP address of the user's local DNS resolver, or other information discussed below.) The Request Router returns a DNS CNAME response by "stacking" the distinguished identifier for Operator B onto the original CDN-domain (e.g., `b.cdn.csp.com`), plus an NS record that maps `b.cdn.csp.com` to B's Request Router.
2. The end-user does a DNS lookup using the modified CDN-domain (i.e., `b.cdn.csp.com`). This causes B's Request Router to respond with a suitable delivery node.
3. The end-user requests the content from B's delivery node. The requested URL contains the name `cdn.csp.com`. (Note that the returned CNAME does not affect the URL.) At this point the

delivery node has the correct IP address of the end-user and can do an HTTP 302 redirect if the redirections in steps 2 and 3 were incorrect. Otherwise B verifies that this CDN-domain belongs to a known peer (so as to avoid being tricked into serving as an open proxy). It then does a DNS request for an "internal" CDN-domain as agreed above (op-b-acq.op-a.net).

4. Operator A recognizes that the DNS request is from a peer CDN rather than an end-user (due to the internal CDN-domain) and so returns the address of a delivery node in uCDN.
5. Operator A serves content to dCDN. Although not shown, it is at this point that Operator A processes the rest of the URL: it extracts information identifying the origin server, validates that this server has been registered, and determines the content provider that owns the origin server.

3.4.1. Comments on the example

The advantages of this approach are that it is more transparent to the end-user and requires fewer round trips than HTTP-based redirection. A potential problem is that the upstream CDN depends on being able to learn the correct downstream CDN that serves the end-user from the client address in the DNS request. In standard DNS operation, uCDN will only obtain the address of the client's local DNS resolver (LDNS), which is not guaranteed to be in the same network (or geographic region) as the client. If not--e.g., the end-user uses a global DNS service--then the upstream CDN cannot determine the appropriate downstream CDN to serve the end-user. In this case, one option is for the upstream CDN to treat the end-user as it would any user not connected to a peer CDN. Another option is for the upstream CDN to "fall back" to a pure HTTP-based redirection strategy in this case (i.e., use the first method). Note that this problem affects existing CDNs that rely on DNS to determine where to redirect client requests, but the consequences are arguably less serious since the LDNS is likely in the same network as the dCDN serves. One approach to ensuring that the client's IP address prefix is correctly determined in such situations is described in [I-D.vandergaast-edns-client-subnet].

As with the prior example, this example partially illustrates the various interfaces involved in CDNI. Operator A could learn dynamically from Operator B the set of prefixes or regions that B is willing and able to serve via the request routing interface. The distinguished name used for acquisition and the identifier for Operator B that is prepended to the CDN domain on redirection are examples of information elements that might also be conveyed by CDNI interfaces (or, alternatively, statically configured). As before,

minimal metadata sufficient to obtain the content is carried "in-band" as part of the redirection process, and standard HTTP is used for inter-CDN acquisition. There is no explicit logging interface discussed in this example.

3.5. Dynamic Footprint Discovery

There could be situations where being able to dynamically discover the set of requests that a given dCDN is willing and able to serve is beneficial. For example, a CDN might at one time be able to serve a certain set of client IP prefixes, but that set might change over time due to changes in the topology and routing policies of the IP network. The following example illustrates this capability. We have chosen the example of DNS-based redirection, but HTTP-based redirection could equally well use this approach.

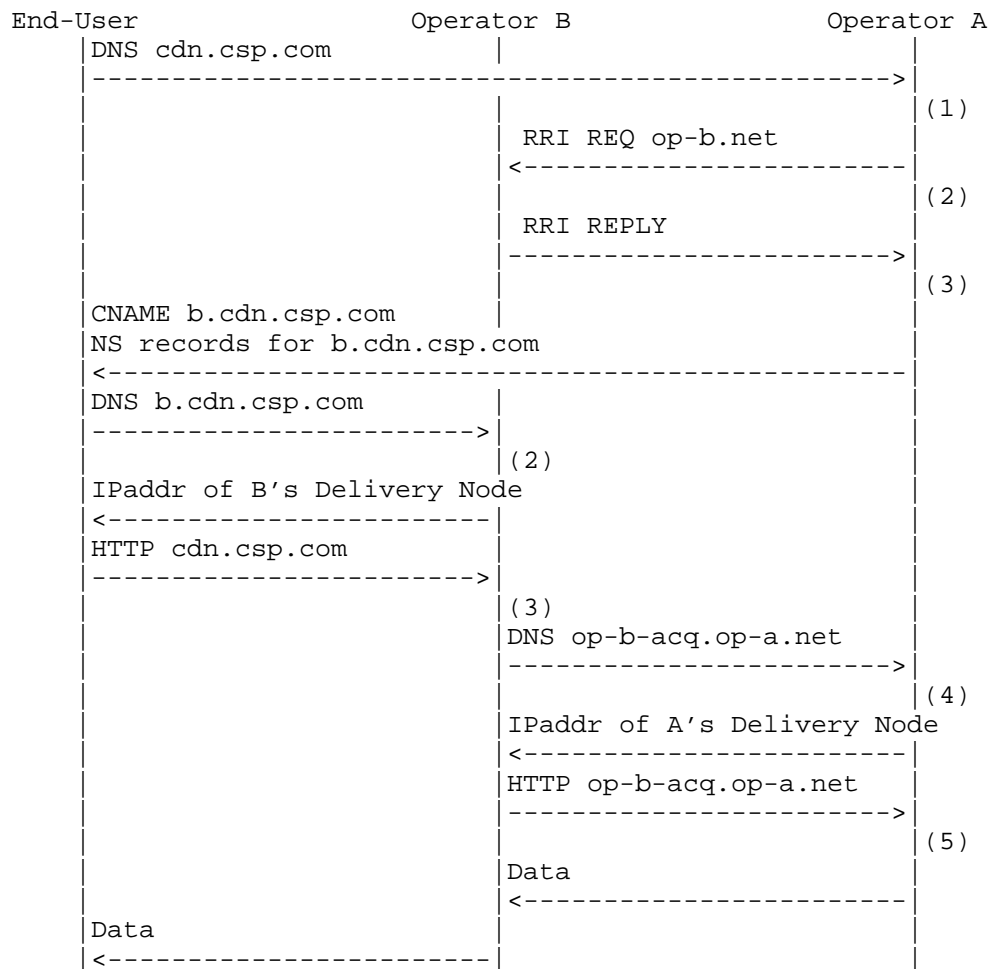


Figure 6: Request Trace for Dynamic Footprint Discovery Example

This example differs from the one in Figure 5 only in the addition of a CDNI Request Routing Interface request (step 2) and corresponding response (step 3). The RRI Req could be a message such as "Can you serve clients from this IP Prefix?" or it could be "Provide the list of client IP prefixes you can currently serve". In either case the response might be cached by operator A to avoid repeatedly asking the same question. Alternatively, or in addition, Operator B may spontaneously advertise to Operator A information (or changes) on the set of requests it is willing and able to serve on behalf of operator A; in that case, Operator B may spontaneously issue RRI REPLY messages that are not in direct response to a corresponding RRI REQ message. (Note that the issues of determining the client's subnet

from DNS requests, as described above, are exactly the same here as in Section 3.4.)

Once Operator A obtains the RRI response, it is now able to determine that Operator B's CDN is an appropriate dCDN for this request and therefore a valid candidate dCDN to consider in its Redirection decision. If that dCDN is selected, the redirection and serving of the request proceeds as before (i.e. in the absence of dynamic footprint discovery).

3.6. Content Removal

The following example illustrates how the Control interface may be used to remove an item of content. In this example, user requests for a particular content, and corresponding redirection of such requests from Operator A to Operator B CDN, may (or may not) have taken place earlier. Then, at some point in time, the uCDN (for example, in response to a corresponding trigger from the Content Provider) uses the Control Interface to request that content identified by a particular URL be removed from dCDN. The following diagram illustrates the operation.

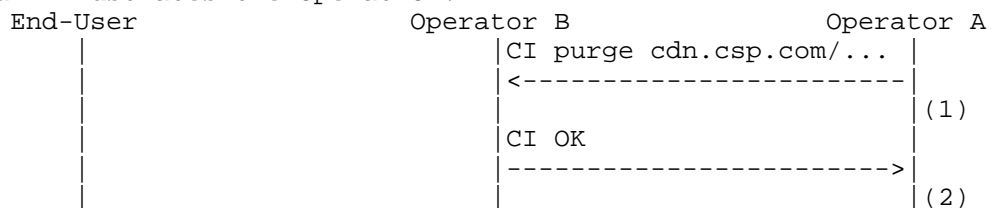


Figure 7: Request Trace for Content Removal

The Control interface is used to convey the request from uCDN to dCDN that some previously acquired content should be deleted. The URL in the request specifies which content to remove. This example corresponds to a DNS-based redirection scenario such as Section 3.4. If HTTP-based redirection had been used, the URL for removal would be of the form peer-a.op-b.net/cdn.csp.com/...

The dCDN is expected to confirm to the uCDN, as illustrated by the CI OK message, the completion of the removal of the targeted content from all the caches in dCDN.

3.7. Pre-Positioned Content Acquisition Example

The following example illustrates how the Control interface may be used to pre-position an item of content in the dCDN. In this example, Operator A uses the Metadata interface to request that

content identified by a particular URL be pre-positioned into Operator B CDN.

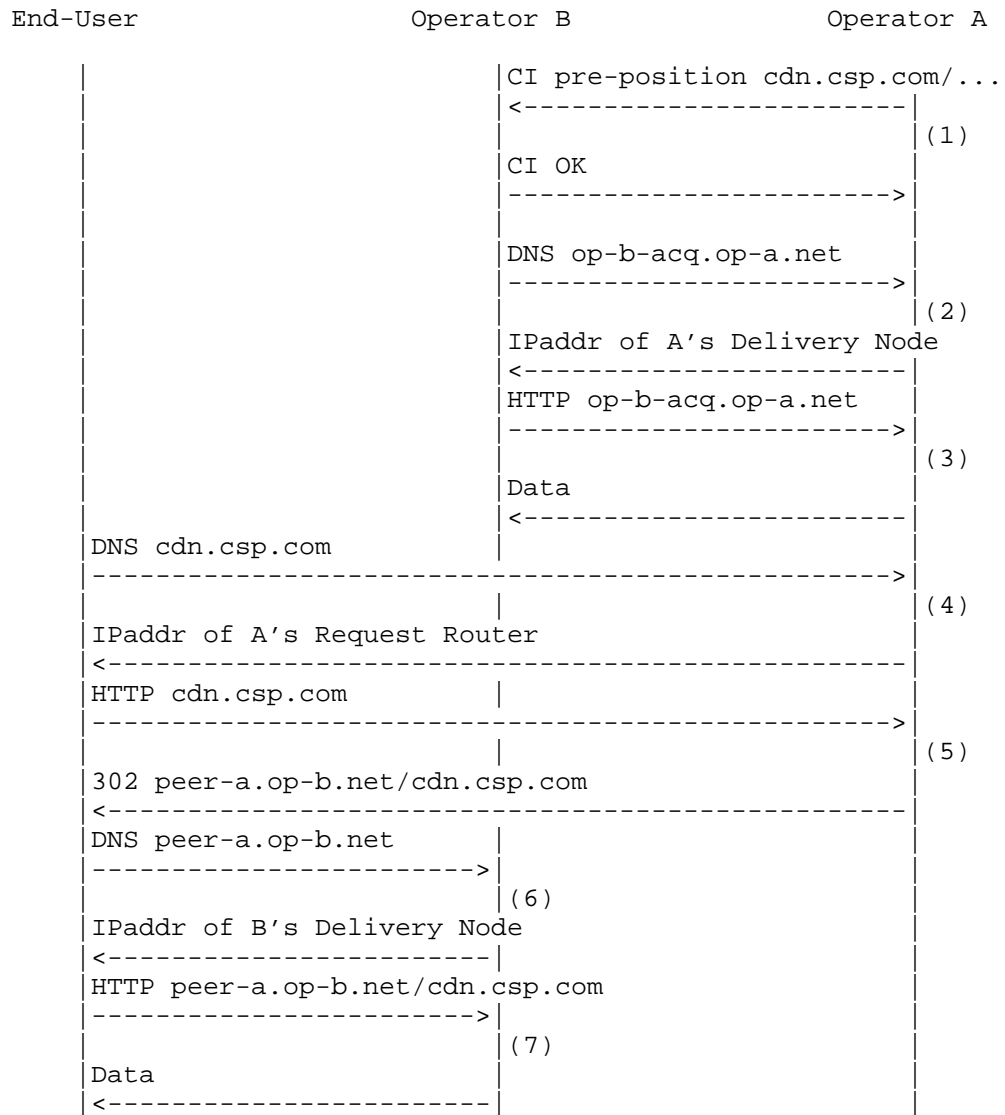


Figure 8: Request Trace for Content Pre-Positioning

The steps illustrated in the figure are as follows:

1. Operator A uses the Control Interface to request that Operator B pre-positions a particular content item identified by its URL.

Operator B responds by confirming that it is willing to perform this operation.

Steps 2 and 3 are exactly the same as steps 5 and 6 of Figure 3, only this time those steps happen as the result of the Pre-positioning request instead of as the result of a cache miss.

Steps 4, 5, 6, 7 are exactly the same as steps 1, 2, 3, 4 of Figure 3, only this time Operator B CDN can serve the end-user request without triggering dynamic content acquisition, since the content has been pre-positioned in dCDN. Note that, depending on dCDN operations and policies, the content pre-positioned in the dCDN may be pre-positioned to all, or a subset of, dCDN caches. In the latter case, intra-CDN dynamic content acquisition may take place inside the dCDN serving requests from caches on which the content has not been pre-positioning; however, such intra-CDN dynamic acquisition would not involve the uCDN.

3.8. Asynchronous CDNI Metadata Example

In this section we walk through a simple example illustrating a scenario of asynchronously exchanging CDNI metadata, where the downstream CDN obtains CDNI metadata for content ahead of a corresponding content request. The example that follows assumes that HTTP-based inter-CDN redirection and recursive CDNI request-routing are used, as in Section 3.3. However, asynchronous exchange of CDNI Metadata is similarly applicable to DNS-based inter-CDN redirection and iterative request routing (in which cases the CDNI metadata may be used at slightly different processing stages of the message flows).

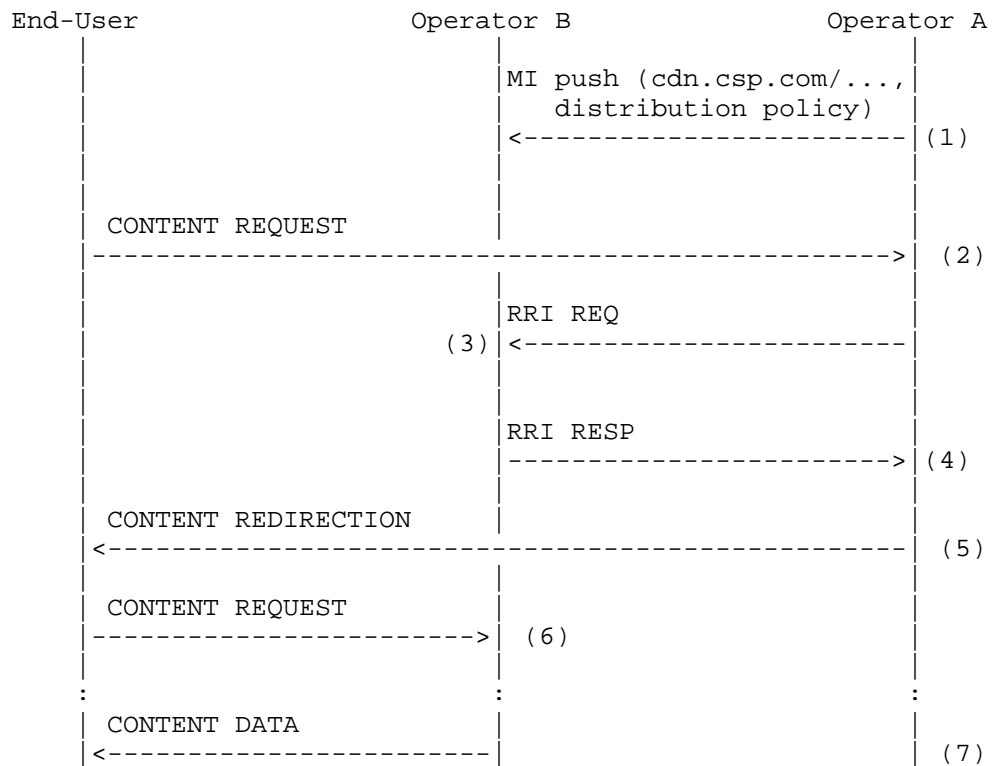


Figure 9: Request Trace for Asynchronous CDNI Metadata

The steps illustrated in the figure are as follows:

- Operator A uses the Metadata Interface to asynchronously push CDNI metadata to Operator B. The present document does not constrain how the CDNI metadata information is actually represented. For the purposes of this example, we assume that Operator A provides CDNI metadata to Operator B indicating that:
 - * this CDNI Metadata is applicable to any content referenced by "cdn.csp.com/op-b.net/..." (assuming HTTP redirection is used - it would be applicable to "cdn.csp.com/..." if DNS redirection were used as in Section 3.4).
 - * this CDNI metadata consists of a distribution policy requiring enforcement by the delivery node of a specific per-request authorization mechanism (e.g. URI signature or token validation).

2. A Content Request occurs as usual.
3. A CDNI Request Routing Request (RRI REQ) is issued by operator A CDN, as discussed in Section 3.3. Operator B's request router can access the CDNI Metadata that are relevant to the requested content and that have been pre-positioned as per Step 1, which may or may not affect the response.
4. Operator B's request router issues a CDNI Request Routing Response (RRI RESP) as in Section 3.3.
5. Operator B performs content redirection as discussed in Section 3.3.
6. On receipt of the Content Request by the end user, the delivery node detects that previously acquired CDNI metadata is applicable to the requested content. In accordance with the specific CDNI metadata of this example, the delivery node will invoke the appropriate per-request authorization mechanism, before serving the content. (Details of this authorization are not shown.)
7. Assuming successful per-request authorization, serving of Content Data (possibly preceded by inter-CDN acquisition) proceeds as in Section 3.3.

3.9. Synchronous CDNI Metadata Acquisition Example

In this section we walk through a simple example illustrating a scenario of synchronous CDNI metadata acquisition, in which the downstream CDN obtains CDNI metadata for content at the time of handling a first request for the corresponding content. As in the preceding section, this example assumes that HTTP-based inter-CDN redirection and recursive CDNI request-routing are used (as in Section 3.3), but dynamic CDNI metadata acquisition is applicable to other variations of request routing.

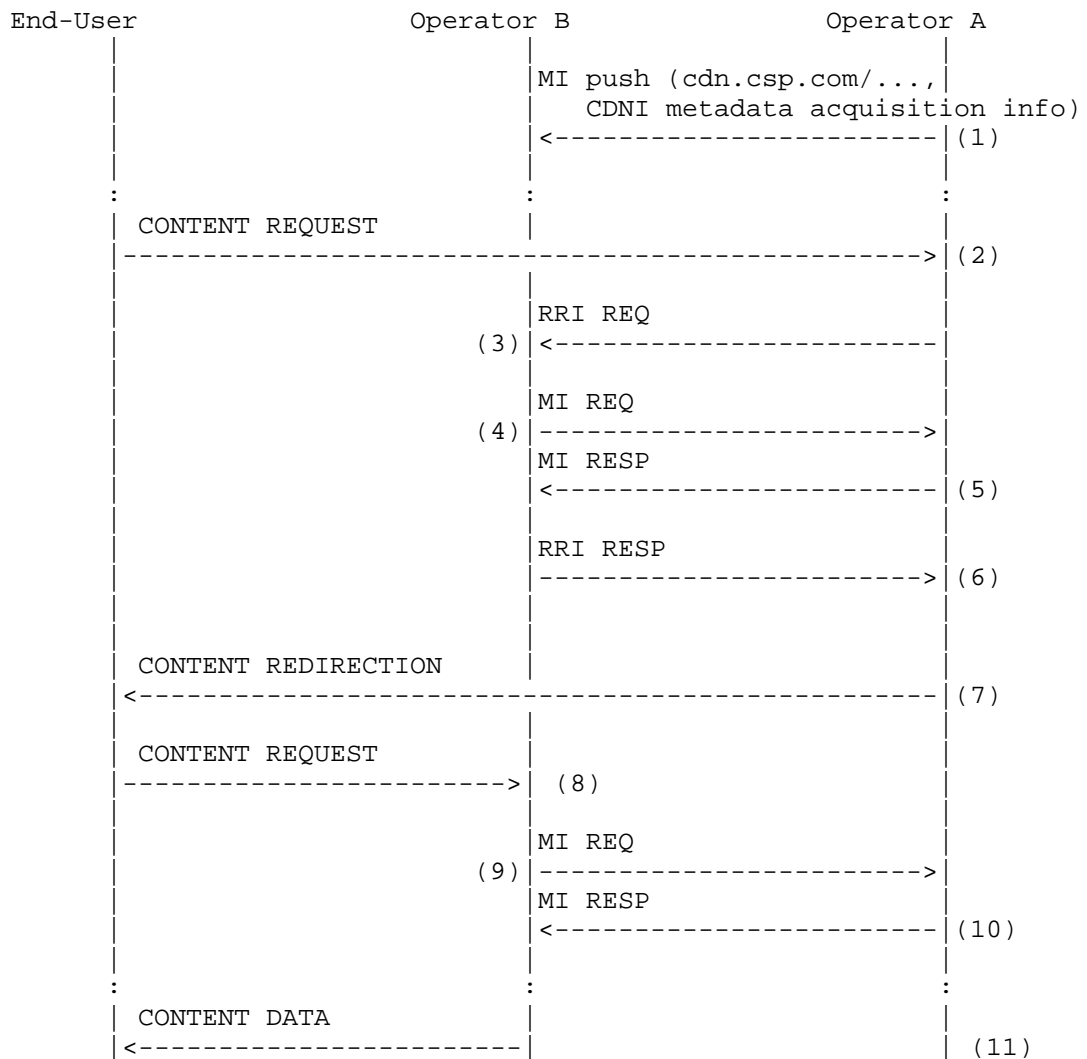


Figure 10: Request Trace for Synchronous CDNI Metadata Acquisition

The steps illustrated in the figure are as follows:

1. Operator A initially uses the Metadata Interface to asynchronously push seed metadata to Operator B. For example, this seed information may include a URI indicating where CDNI Metadata can later be pulled from for some content set. (There are alternative ways that this seeding information may be provided, such as piggybacking on the CDNI RRI REQ message of

Step 3.)

2. A Content Request arrives as normal.
3. A Request Routing Interface request occurs as in the prior example.
4. On receipt of the CDNI Request Routing Request, Operator B's CDN initiates synchronous acquisition of CDNI Metadata that are needed for routing of the end-user request. The seeding information provided in Step 1 is used to determine how to obtain the metadata. Note that there may exist cases in which this step does not occur (e.g., because the CDNI metadata seeding information indicates CDNI metadata are not needed at that stage).
5. On receipt of a CDNI Metadata MI Request, Operator A's CDN responds, making the corresponding CDNI metadata information available to Operator B's CDN. This metadata is considered by operator B's CDN before responding to the Request Routing request. (In a simple case, the metadata could simply be an allow or deny response for this particular request.)
6. Response to the RRI request as normal.
7. Redirection message is sent to the end user.
8. A delivery node of Operator B receives the end user request.
9. The delivery node triggers dynamic acquisition of additional CDNI metadata that are needed to process the end-user content request. Again the seeding information provided in Step 1 is used to determine how to acquire the needed CDNI metadata. Note that there may exist cases where this step need not happen, either because the metadata were already acquired previously, or because the seeding information indicates no metadata are required.
10. Operator A's CDN responds to the CDNI Metadata Request and makes the corresponding CDNI metadata available to Operator B. This metadata influence how Operator B's CDN processes the end-user request.
11. Content is served (possibly preceded by inter-CDN acquisition) as in Section 3.3.

3.10. Content Acquisition with Multiple Upstream CDNs

A single dCDN may receive end-user requests from multiple uCDNs. When a dCDN receives an end-user request, it must determine the identity of the uCDN from which it should acquire the requested content.

Ideally, the acquisition path of an end-user request will follow the redirection path of the request. The dCDN should acquire the content from the same uCDN which redirected the request.

Determining the acquisition path requires the dCDN to reconstruct the redirection path based on information in the end-user request. The method for reconstructing the redirection path differs based on the redirection approach: HTTP or DNS.

With HTTP-redirection, the rewritten URI must include sufficient information for the dCDN to directly or indirectly determine the uCDN when the end-user request is received. The HTTP-redirection approach can be further broken-down based on how the URL is rewritten during redirection: HTTP-redirection with or without Site Aggregation. HTTP-redirection with Site Aggregation hides the identity of the original CSP. HTTP-redirection without Site Aggregation does not attempt to hide the identity of the original CSP. With both approaches, the rewritten URI must include enough information to identify the immediate neighbor uCDN.

With DNS-redirection, the dCDN receives the published URI (instead of a rewritten URI) and does not have sufficient information for the dCDN to identify the appropriate uCDN. Given that the dCDN has established a business relationship with each of its uCDNs, assume that the dCDN can trust any uCDN to acquire the content. The dCDN may narrow the set of viable uCDNs by examining the CDNI metadata from each to determine which uCDNs are hosting metadata for the requested content.

Content acquisition may be preceded by content metadata acquisition. If possible, the acquisition path for metadata should also follow the redirection path. Additionally, metadata must be indexed based on rewritten URIs in the case of HTTP-redirection and must be indexed based on published URIs in the case of DNS-redirection. Thus, the request routing interface and the metadata interface are tightly coupled in that the result of request routing (a rewritten URI pointing to the dCDN) must serve as an input to metadata lookup. If the content metadata includes information for acquiring the content, then the metadata interface is also tightly coupled with the acquisition interface in that the result of the metadata lookup (an acquisition URL likely hosted by the uCDN) should serve as input to

the content acquisition.

4. Main Interfaces

Figure 1 illustrates the four main interfaces that are in scope for the CDNI WG, along with several others. The detailed specifications of these interfaces are left to other documents (mostly still to be written, but see [I-D.ietf-cdni-problem-statement] and [I-D.ietf-cdni-requirements] for some discussion of the interfaces).

One interface that is not shown in Figure 1 is the interface between the user and the CSP. While for the purposes of CDNI that interface is out of scope, it is worth noting that it does exist and can provide useful functions, such as end-to-end performance monitoring and some forms of authentication and authorization.

There is also an important interface between the user and the Request Routing function of both uCDN and dCDN. As we saw in some of the preceding examples, that interface can be used as a way of passing information such as the metadata that is required to obtain the content in dCDN from uCDN.

In this section we will provide an overview of the functions performed by each of the CDNI interfaces and discuss how they fit into the overall solution. We also examine some of the design tradeoffs. We begin with an examination of one such tradeoff that affects all the interfaces - the use of in-band or out-of-band communication.

4.1. In-Band versus Out-of-Band Interfaces

Before getting to the individual interfaces, we observe that there is a high-level design choice for each, involving the use of existing in-band communication channels versus defining new out-of-band interfaces.

It is possible that the information needed to carry out various interconnection functions can be communicated between peer CDNs using existing in-band protocols. The use of HTTP 302 redirect is an example of how certain aspects of request routing can be implemented in-band (embedded in URIs). Note that using existing in-band protocols does not imply that the CDNI interfaces are null; it is still necessary to establish the rules (conventions) by which such protocols are used to implement the various interface functions.

There are other opportunities for in-band communication beyond HTTP redirects. For example, many of the HTTP directives used by proxy

servers can also be used by peer CDNs to inform each other of caching activity. Of these, one that is particularly relevant is the If-Modified-Since directive, which is used with the GET method to make it conditional: if the requested object has not been modified since the time specified in this field, a copy of the object will not be returned, and instead, a 304 (not modified) response will be returned.

4.2. Cross Interface Concerns

Although the CDNI interfaces are largely independent, there are a set of conventions practiced consistently across all interfaces. Most important among these is how resources are named, for example, how the Metadata and Control interfaces identify the set of resources to which a given directive applies, or the Logging interface identifies the set of resources for which a summary record applies.

While in the limit the CDNI interfaces could explicitly identify every individual resource, in practice, they name resource aggregates (sets of URIs) that are to be treated in a similar way. For example, URI aggregates can be identified by a CDN-Domain (i.e., the FQDN at the beginning of a URI) or by a URI-Filter (i.e., a regular expression that matches a subset of URIs contained in some CDN-Doman). In other words, CDN-Domains and URI-Filters provide a uniform means to aggregate sets (and subsets) of URIs for the purpose of defining the scope for some operation in one of the CDNI interfaces.

4.3. Request Routing Interface

We may think of the request routing interface as comprising two parts: the asynchronous advertisement of footprint and capabilities by a dCDN that allows a uCDN to decide whether to redirect particular user requests to that dCDN; and the synchronous operation of actually redirecting a user request. (These are somewhat analogous to the operations of routing and forwarding in IP.)

As illustrated in Section 3, the synchronous part of the request routing interface may be implemented in part by DNS and HTTP. Naming conventions may be established by which CDN peers communicate whether a request should be routed or content served.

In support of these exchanges, it is necessary for CDN peers to exchange additional information with each other. Depending on the method(s) supported, this includes

- o The operator's unique id (operator-id) or distinguished CDN-domain (operator-domain);

- o NS records for the operator's set of externally visible request routers;
- o The set of requests the dCDN operator is prepared to serve (e.g. a set of client IP prefixes or geographic regions that may be served by dCDN).

Of these, the two operator identifiers are fixed, and can be exchanged off-line as part of a peering agreement. The NS records potentially change with some frequency, but an existing protocol--DNS--can be used to dynamically track this information. That is, a peer can do a DNS lookup on operator-domain to retrieve the set of NS records corresponding to the peer's redirection service.

The set of requests that dCDN is willing to serve could in some cases be relatively static (e.g., a set of IP prefixes) which could be exchanged off-line, or might even be negotiated as part of a peering agreement. However, it may also be more dynamic, in which case an explicit protocol for its exchange would be helpful.

A variety of options exist for the dCDN operator to advertise its footprint to uCDN. As discussed in [I-D.previdi-cdni-footprint-advertisement], footprint is comprised of two components:

- o a class of end user requests (represented, for example, by a set of IP prefixes, or a geographic region) that the dCDN is willing and able to serve directly, without use of another dCDN;
- o the connectivity of the dCDN to other CDNs that may be able to serve content to users on behalf of dCDN.

[I-D.previdi-cdni-footprint-advertisement] describes an approach to advertising such footprint information asynchronously using BGP. In addition to this sort of information, a dCDN might also advertise "capabilities" such as the ability to handle certain types of content (e.g. specific streaming formats) or quality of service (QoS) capabilities. [I-D.xiaoyan-cdni-request-routing-protocol] describes an approach that exchanges CDN "capabilities" over HTTP, while [I-D.seedorf-alto-for-cdni] describes how ALTO [RFC5693] may be used to obtain request routing information.

We also note that the Request Routing interface plays a key role in enabling recursive redirection, as illustrated in Section 3.3. It enables the user to be redirected to the correct delivery node in dCDN with only a single redirection step (as seen by the user). This may be particularly valuable as the chain of interconnected CDNs increases beyond two CDNs.

One final issue is how HTTP adaptive streaming impacts the Request Routing interface, and in particular, the interplay between the request router and manifest files, which may contain either absolute or relative URLs. These issues are discussed in more detail in [I-D.brandenburg-cdni-has].

4.4. Logging Interface

It is necessary for the upstream CDN to have visibility into the delivery of content it originates to end-users connected to the downstream CDN. This allows the upstream CDN to properly bill its customers for multiple deliveries of content cached by the downstream CDN, as well as to report accurate traffic statistics to those content providers. This is one role of the Logging interface.

Other operational data that may be relevant to CDNI can also be exchanged by the Logging interface. For example, dCDN may report the amount of content it has acquired from uCDN, and how much cache storage has been consumed by content cached on behalf of uCDN.

Traffic logs are easily exchanged off-line. For example, the following traffic log is a small deviation from the Apache log file format, where entries include the following fields:

- o Domain - the full domain name of the origin server
- o IP address - the IP address of the client making the request
- o End time - the ending time of the transfer
- o Time zone - any time zone modifier for the end time
- o Method - the transfer command itself (e.g., GET, POST, HEAD)
- o URL - the requested URL
- o Version - the protocol version, such as HTTP/1.0
- o Response - a numeric response code indicating transfer result
- o Bytes Sent - the number of bytes in the body sent to the client
- o Request ID - a unique identifier for this transfer
- o User agent - the user agent, if supplied
- o Duration - the duration of the transfer in milliseconds

- o Cached Bytes - the number of body bytes served from the cache
- o Referrer - the referrer string from the client, if supplied

Of these, only the Domain field is indirect in the downstream CDN--it is set to the CDN-domain used by the upstream CDN rather than the actual origin server. This field could then be used to filter traffic log entries so only those entries matching the upstream CDN are reported to the corresponding operator. Further discussion of the Logging interface can be found in [I-D.bertrand-cdni-logging] and [I-D.lefaucheur-cdni-logging-delivery].

One open question is who does the filtering. One option is that the downstream CDN filters its own logs, and passes the relevant records directly to each upstream peer. This requires that the downstream CDN knows the set of CDN-domains that belong to each upstream peer. If this information is already exchanged between peers as part of the request routing interface, then direct peer-to-peer reporting is straightforward. If it is not available, and operators do not wish to advertise the set of CDN-domains they serve to their peers, then the second option is for each CDN to send both its non-local traffic records and the set of CDN-domains it serves to an independent third-party (i.e., a CDN Exchange), which subsequently filters, merges, and distributes traffic records on behalf of each participating CDN operator.

A second open question is how timely traffic information should be. For example, in addition to off-line traffic logs, accurate real-time traffic monitoring might also be useful, but such information requires that the downstream CDN inform the upstream CDN each time it serves upstream content from its cache. The downstream CDN can do this, for example, by sending a conditional HTTP GET request (If-Modified-Since) to the upstream CDN each time it receives an HTTP GET request from one of its end-users. This allows the upstream CDN to record that a request has been issued for the purpose of real-time traffic monitoring. The upstream CDN can also use this information to validate the traffic logs received later from the downstream CDN.

There is obviously a tradeoff between accuracy of such monitoring and the overhead of the downstream CDN having to go back to the upstream CDN for every request.

Another design tradeoff in the Logging interface is the degree of aggregation or summarization of data. One situation that lends itself to summarization is the delivery of HTTP-based adaptive bit-rate video. Most schemes to deliver such video use a large number of relatively small HTTP requests (e.g. one request per two-second chunk of video.) It may be desirable to aggregate logging information so

that a single log entry is provided for the entire video rather than for each chunk. Note however that such aggregation requires a degree of application awareness in dCDN to recognize that the many HTTP requests correspond to a single video. The implications of HTTP adaptive streaming are discussed further in [I-D.brandenburg-cdni-has].

Other forms of aggregation may also be useful. For example, there may be situations where bulk metrics such as bytes delivered per hour may suffice rather than the detailed per-request logs outlined above. It seems likely that a range of granularities of logging will be needed along with ways to specify the type and degree of aggregation required.

4.5. Control Interface

The control interface is initially used to bootstrap the other interfaces. As a simple example, it could be used to provide the address of the logging server in dCDN to uCDN in order to bootstrap the logging interface. It may also be used, for example, to establish security associations for the other interfaces.

The other role the Control interface plays is to allow the uCDN to pre-position, revalidate, or purge metadata and content on a dCDN. These operations, sometimes collectively called the trigger interface, are discussed further in [I-D.murray-cdni-triggers].

4.6. Metadata Interface

The role of the metadata interface is to enable CDNI distribution metadata to be conveyed to the downstream CDN by the upstream CDN. For example, see [I-D.ma-cdni-metadata] and [I-D.cjlmw-cdni-metadata]. Such metadata includes geo-blocking restrictions, availability windows, access control policies, and so on. It may also include policy information such as the desire to pre-position content rather than fetch it on demand.

Some metadata may be able to be conveyed using in-band mechanisms. For example, to inform the downstream CDN of any geo-blocking restrictions or availability windows, the upstream can elect to redirect a request to the downstream CDN only if that CDN's advertised delivery footprint is acceptable for the requested URL. Similarly, the request could be forwarded only if the current time is within the availability window.

Similarly, some forms of access control may also be performed on a per-request basis using HTTP directives. For example, being able to respond to a conditional GET request gives the upstream CDN an

opportunity to influence how the downstream CDN delivers its content. Minimally, the upstream CDN can invalidate (purge) content previously cached by the downstream CDN.

Fine-grain control over how the downstream CDN delivers content on behalf of the upstream CDN is also possible. For example, by including the X-Forwarded-For HTTP header with the conditional GET request, the downstream CDN can report the end-user's IP address to the upstream CDN, giving it an opportunity to control whether the downstream CDN should serve the content to this particular end-user. The upstream CDN would communicate its directive through its response to the conditional GET. The downstream CDN can cache information for a period of time specified by the upstream CDN, thereby reducing control overhead.

All of these in-band techniques serve to illustrate that uCDNs have the option of enforcing their access control policies themselves, rather than delegating enforcement to dCDNs using the Metadata interface. As a consequence, the Metadata interface must provide a means for the uCDN to express this its desire to retain enforcement for itself. For example, this might be done by including a "check with me" flag in the metadata associated with certain content.

5. Deployment Models

In this section we describe a number of possible deployment models that may be achieved using the CDNI interfaces described above. We note that these models are by no means exhaustive, and that many other models may be possible.

Although the reference model of Figure 1 shows all CDN functions on each side of the CDNI interface, deployments can rely on entities that are involved in any subset of these functions, and therefore only support the relevant subset of CDNI interfaces. As already noted in Section 3, effective CDNI deployments can be built without necessarily implementing all four interfaces. Some examples of such deployments are shown below.

Note that, while we refer to upstream and downstream CDNs, this distinction applies to specific content items and transactions. That is, a given CDN may be upstream for some transactions and downstream for others, depending on many factors such as location of the requesting client and the particular piece of content requested.

5.1. Meshed CDNs

Although the reference model illustrated in Figure 1 shows a unidirectional CDN interconnection with a single uCDN and a single dCDN, any arbitrary CDNI meshing can be built from this, such as the example meshing illustrated in Figure 11. (Support for arbitrary meshing may or may not be in the initial scope for the working group, but the model allows for it.)

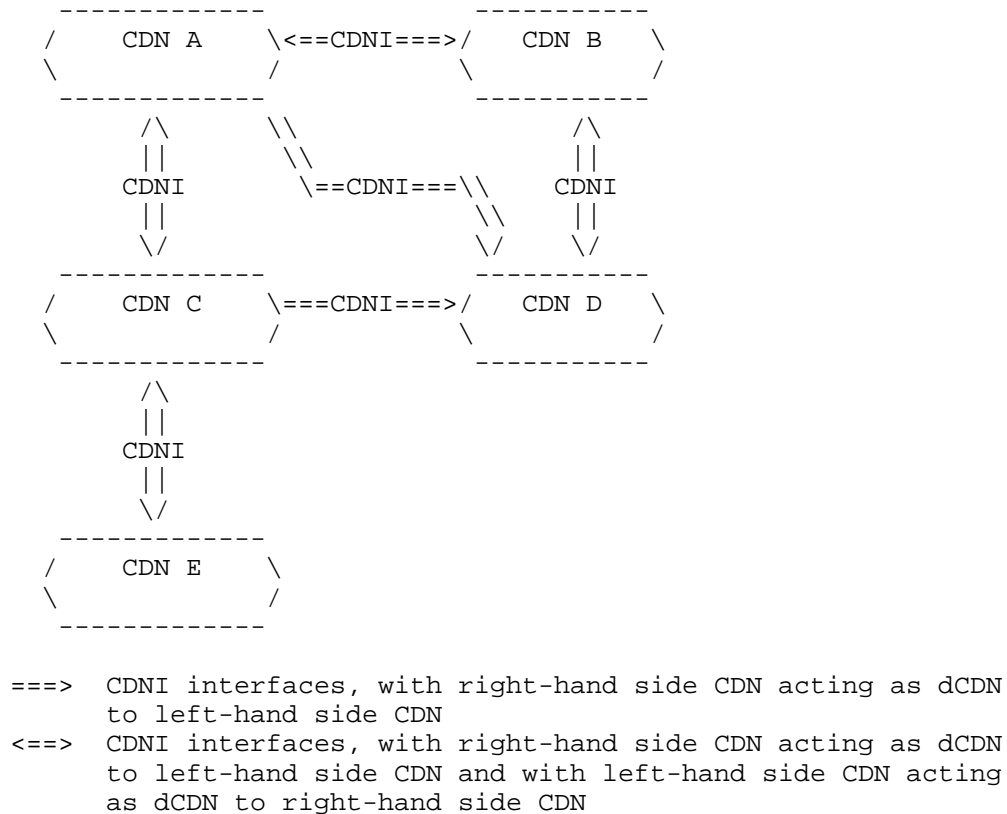


Figure 11: CDNI Deployment Model: CDN Meshing Example

5.2. CSP combined with CDN

Note that our terminology refers to functional roles and not economic or business roles. That is, a given organization may be operating as both a CSP and a fully-fledged uCDN when we consider the functions performed, as illustrated in Figure 12.

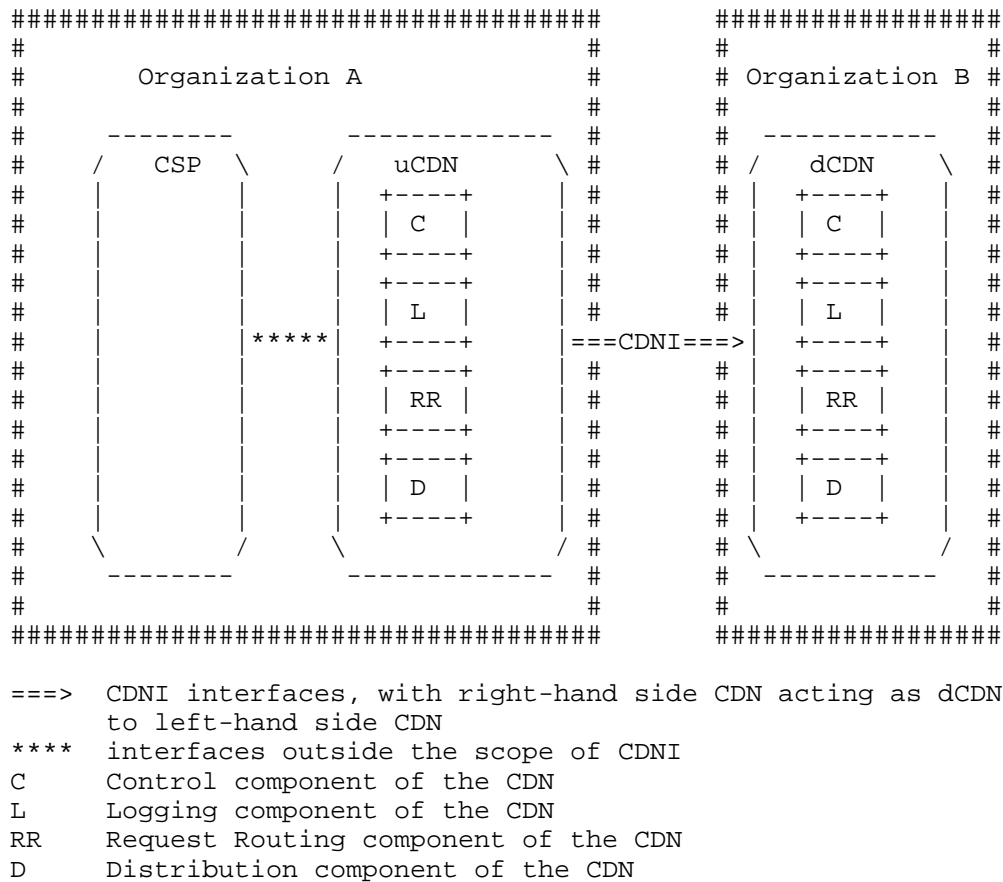


Figure 12: CDNI Deployment Model: Organization combining CSP & uCDN

5.3. CSP using CDNI Request Routing Interface

As another example, a content provider organization may choose to run its own request routing function as a way to select among multiple candidate CDN providers; In this case the content provider may be modeled as the combination of a CSP and of a special, restricted case of a CDN. In that case, as illustrated in Figure 13, the CDNI Request Routing interface can be used between the restricted CDN operated by the content provider Organization and the CDN operated by the full-CDN organization acting as a dCDN in the request routing control plane. Interfaces outside the scope of the CDNI work can be used between the CSP functional entities of the content provider organization and the CDN operated by the full-CDN organization acting as a uCDN) in the CDNI control planes other than the request routing plane (i.e. Control, Distribution, Logging).

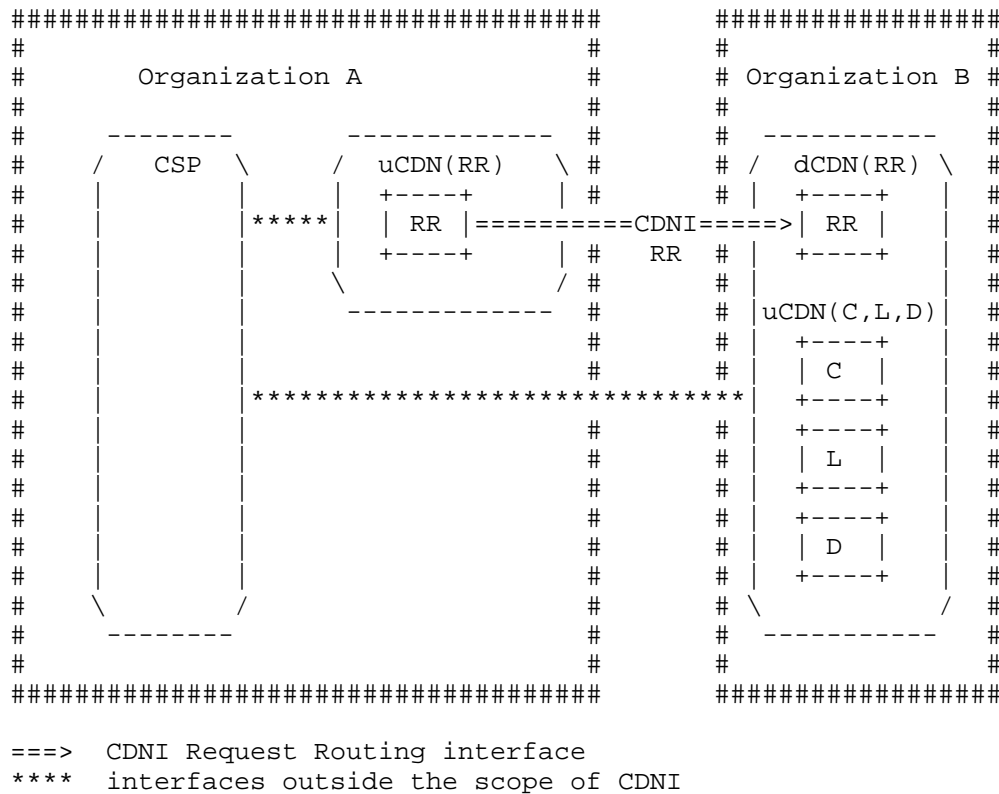


Figure 13: CDNI Deployment Model: Organization combining CSP and partial CDN

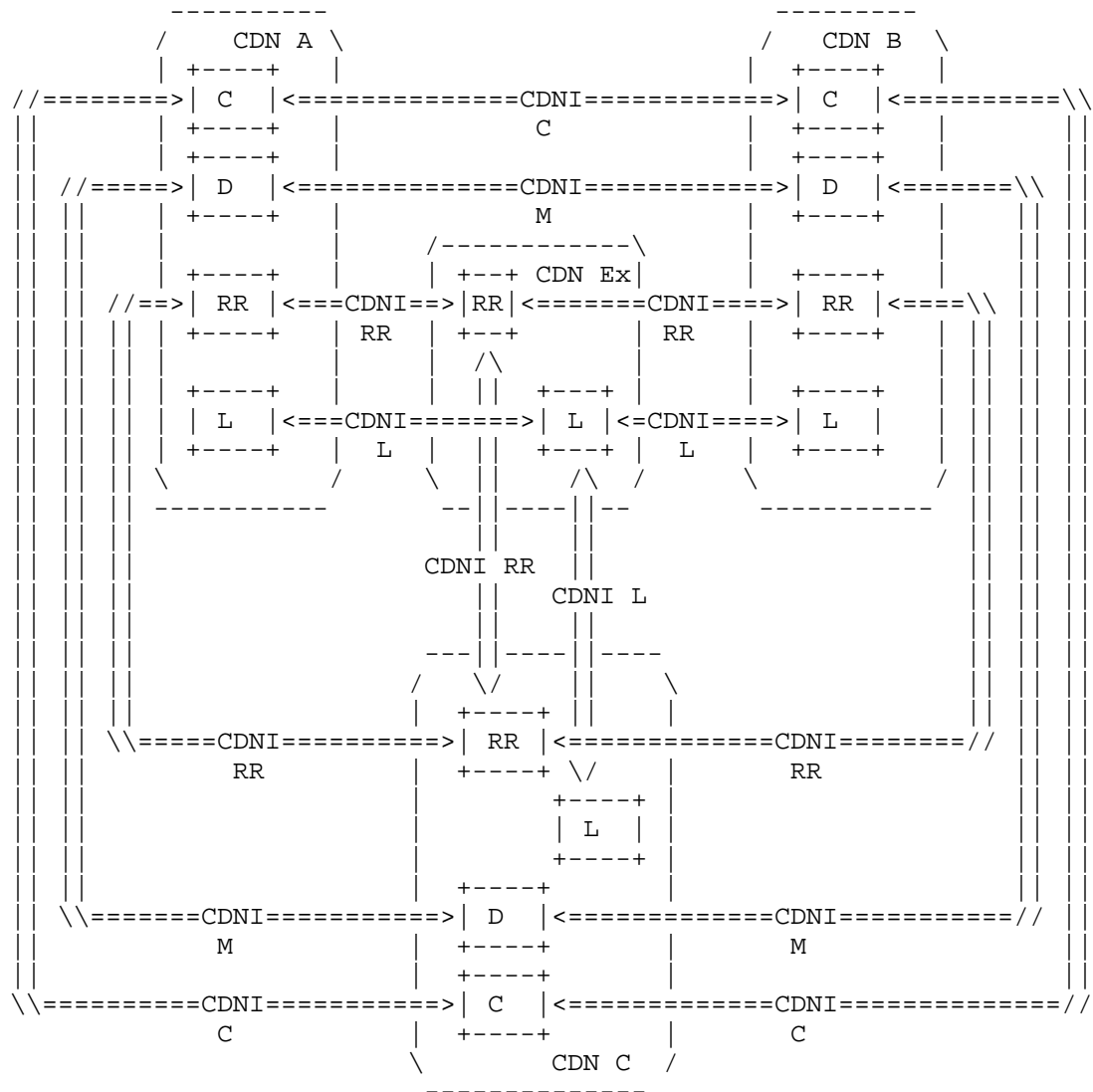
5.4. CDN Federations and CDN Exchanges

There are two additional concepts related to, but distinct from CDN Interconnection. The first is CDN Federation. Our view is that CDNI is the more general concept, involving two or more CDNs serving content to each other's users, while federation implies a multi-lateral interconnection arrangement, but other CDN interconnection agreements are also possible (e.g., symmetric bilateral, asymmetric bilateral). An important conclusion is that CDNI technology should not presume (or bake in) a particular interconnection agreement, but should instead be general enough to permit alternative interconnection arrangements to evolve.

The second concept often used in the context of CDN Federation is CDN Exchange--a third party broker or exchange that is used to facilitate a CDN federation. Our view is that a CDN exchange offers valuable machinery to scale the number of CDN operators involved in a multi-

lateral (federated) agreement, but that this machinery is built on top of the core CDNI interconnection mechanisms. For example, as illustrated in Figure 14, the exchange might aggregate and redistribute information about each CDN footprint and capacity, as well as collect, filter, and re-distribute traffic logs that each participant needs for interconnection settlement, but inter-CDN request routing, inter-CDN content distribution (including inter-CDN acquisition) and inter-CDN control which fundamentally involve a direct interaction between an upstream CDN and a downstream CDN-- operate exactly as in a pair-wise peering arrangement. Turning to Figure 14, we observe that in this example:

- o each CDN supports a direct CDNI Control interface to every other CDN
- o each CDN supports a direct CDNI Metadata interface to every other CDN
- o each CDN supports a CDNI Logging interface with the CDN Exchange
- o each CDN supports both a CDNI request Routing interface with the CDN Exchange (for aggregation and redistribution of dynamic CDN footprint discovery information) and a direct CDNI Request Routing interface to every other CDN (for actual request redirection).



<=CDNI RR=> CDNI Request Routing interface
 <=CDNI M==> CDNI Metadata interface
 <=CDNI C==> CDNI Control interface
 <=CDNI L==> CDNI Logging interface

Figure 14: CDNI Deployment Model: CDN Exchange

Note that a CDN exchange may alternatively support a different set of functionality (e.g. Logging only, or Logging and full request

routing, or all the functionality of a CDN including content distribution). All these options are expected to be allowed by the IETF CDNI specifications.

6. Trust Model

There are a number of trust issues that need to be addressed by a CDNI solution. Many of them are in fact similar or identical to those in a simple CDN without interconnection. In a standard CDN environment (without CDNI), the CSP places a degree of trust in a single CDN operator to perform many functions. The CDN is trusted to deliver content with appropriate quality of experience for the end user. The CSP trusts the CDN operator not to corrupt or modify the content. The CSP often relies on the CDN operator to provide reliable accounting information regarding the volume of delivered content. The CSP may also trust the CDN operator to perform actions such as timely invalidation of content and restriction of access to content based on certain criteria such as location of the user and time of day, and to enforce per-request authorization performed by the CSP using techniques such as URI signing.

A CSP also places trust in the CDN not to distribute any information that is confidential to the CSP (e.g., how popular a given piece of content is) or confidential to the end user (e.g., which content has been watched by which user).

A CSP does not necessarily have to place complete trust in a CDN. A CSP will in some cases take steps to protect its content from improper distribution by a CDN, e.g. by encrypting it and distributing keys in some out of band way. A CSP also depends on monitoring (possibly by third parties) and reporting to verify that the CDN has performed adequately. A CSP may use techniques such as client-based metering to verify that accounting information provided by the CDN is reliable. HTTP conditional requests may be used to provide the CSP with some checks on CDN operation. In other words, while a CSP may trust a CDN to perform some functions in the short term, the CSP is able in most cases to verify whether these actions have been performed correctly and to take action (such as moving the content to a different CDN) if the CDN does not live up to expectations.

The main trust issue raised by CDNI is that it introduces transitive trust. A CDN that has a direct relationship with a CSP can now "outsource" the delivery of content to another (downstream) CDN. That CDN may in turn outsource delivery to yet another downstream CDN, and so on.

The top level CDN in such a chain of delegation is responsible for ensuring that the requirements of the CSP are met. Failure to do so is presumably just as serious as in the traditional single CDN case. Hence, an upstream CDN is essentially trusting a downstream CDN to perform functions on its behalf in just the same way as a CSP trusts a single CDN. Monitoring and reporting can similarly be used to verify that the downstream CDN has performed appropriately. However, the introduction of multiple CDNs in the path between CSP and end user complicates the picture. For example, third party monitoring of CDN performance (or other aspects of operation, such as timely invalidation) might be able to identify the fact that a problem occurred somewhere in the chain but not point to the particular CDN at fault.

In summary, we assume that an upstream CDN will invest a certain amount of trust in a downstream CDN, but that it will verify that the downstream CDN is performing correctly, and take corrective action (including potentially breaking off its relationship with that CDN) if behavior is not correct. We do not expect that the trust relationship between a CSP and its "top level" CDN will differ significantly from that found today in single CDN situations. However, it does appear that more sophisticated tools and techniques for monitoring CDN performance and behavior will be required to enable the identification of the CDN at fault in a particular delivery chain.

We expect that the detailed designs for the specific interfaces for CDNI will need to take the transitive trust issues into account. For example, explicit confirmation that some action (such as content removal) has taken place in a downstream CDN may help to mitigate some issues of transitive trust.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

While there is a variety of security issues introduced by a single CDN, we are concerned here specifically with the additional issues that arise when CDNs are interconnected. For example, when a single CDN has the ability to distribute content on behalf of a CSP, there may be concerns that such content could be distributed to parties who are not authorized to receive it, and there are mechanisms to deal with such concerns. Our focus in this section is on how CDN interconnection introduces new security issues not found in the

single CDN case.

Many of the security issues that arise in CDNI are related to the transitivity of trust (or lack thereof) described in Section 6. As noted above, the design of the various interfaces for CDNI must take account of the additional risks posed by the fact that a CDN with whom a CSP has no direct relationship is now potentially distributing content for that CSP. The mechanisms used to mitigate these risks may be similar to those used in the single CDN case, but their suitability in this more complex environment must be validated.

Another concern that arises in any CDN is that information about the behavior of users (what content they access, how much content they consume, etc.) may be gathered by the CDN. This risk certainly exists in inter-connected CDNs, but it should be possible to apply the same techniques to mitigate it as in the single CDN case.

CDNs today offer a variety of means to control access to content, such as time-of-day restrictions, geo-blocking, and URI signing. These mechanisms must continue to function in CDNI environments, and this consideration is likely to affect the design of certain CDNI interfaces (e.g. metadata, request routing.)

Just as with a single CDN, each peer CDN must ensure that it is not used as an "open proxy" to deliver content on behalf of a malicious CSP. Whereas a single CDN typically addresses this problem by having CSPs explicitly register content (or origin servers) that is to be served, simply propagating this information to peer downstream CDNs may be problematic because it reveals more information than the upstream CDN is willing to specify. (To this end, the content acquisition step in the earlier examples force the dCDN to retrieve content from the uCDN rather than go directly to the origin server.)

There are several approaches to this problem. One is for the uCDN to encode a signed token generated from a shared secret in each URL routed to a dCDN, and for the dCDN to validate the request based on this token. Another one is to have each upstream CDN advertise the set of CDN-domains they serve, where the downstream CDN checks each request against this set before caching and delivering the associated object. Although straightforward, this approach requires operators to reveal additional information, which may or may not be an issue.

8.1. Security of CDNI Interfaces

It is noted in [I-D.ietf-cdni-requirements] that all CDNI interfaces must be able to operate securely over insecure IP networks. Since it is expected that the CDNI interfaces will be implemented using existing application protocols such as HTTP or XMPP, we also expect

that the security mechanisms available to those protocols may be used by the CDNI interfaces. Details of how these interfaces are secured will be specified in the relevant interface documents.

8.2. Digital Rights Management

Issues of digital rights management (DRM, also sometimes called digital restrictions management) is often employed for content distributed via CDNs. In general, DRM relies on the CDN to distribute encrypted content, with decryption keys distributed to users by some other means (e.g. directly from the CSP to the end user.) For this reason, DRM is considered out of scope for the CDNI WG [I-D.ietf-cdni-problem-statement] and does not introduce additional security issues for CDNI.

9. Contributors

The following individuals contributed to this document:

- o Matt Caulfield
- o Francois le Faucheur
- o Aaron Falk
- o David Ferguson
- o John Hartman
- o Ben Niven-Jenkins
- o Kent Leung

10. Acknowledgements

We thank Huw Jones for helpful input to the draft.

11. Informative References

[I-D.bertrand-cdni-logging]
Bertrand, G. and S. Emile, "CDNI Logging Interface",
draft-bertrand-cdni-logging-00 (work in progress),
February 2012.

[I-D.brandenburg-cdni-has]

Brandenburg, R., Deventer, O., Faucheur, F., and K. Leung, "Models for adaptive-streaming-aware CDN Interconnection", draft-brandenburg-cdni-has-03 (work in progress), July 2012.

[I-D.cjlmw-cdni-metadata]

Niven-Jenkins, B., Murray, R., Watson, G., Caulfield, M., and K. Leung, "CDN Interconnect Metadata", draft-cjlmw-cdni-metadata-00 (work in progress), July 2012.

[I-D.ietf-cdni-problem-statement]

Niven-Jenkins, B., Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", draft-ietf-cdni-problem-statement-08 (work in progress), June 2012.

[I-D.ietf-cdni-requirements]

Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", draft-ietf-cdni-requirements-03 (work in progress), June 2012.

[I-D.ietf-cdni-use-cases]

Bertrand, G., Emile, S., Burbridge, T., Eardley, P., Ma, K., and G. Watson, "Use Cases for Content Delivery Network Interconnection", draft-ietf-cdni-use-cases-09 (work in progress), July 2012.

[I-D.lefaucheur-cdni-logging-delivery]

Faucheur, F., Viveganandhan, M., and K. Leung, "CDNI Logging Formats for HTTP and HTTP Adaptive Streaming Deliveries", draft-lefaucheur-cdni-logging-delivery-01 (work in progress), July 2012.

[I-D.ma-cdni-metadata]

Ma, K., "Content Distribution Network Interconnection (CDNI) Metadata Interface", draft-ma-cdni-metadata-03 (work in progress), July 2012.

[I-D.murray-cdni-triggers]

Murray, R. and B. Niven-Jenkins, "CDN Interconnect Triggers", draft-murray-cdni-triggers-00 (work in progress), February 2012.

[I-D.previdi-cdni-footprint-advertisement]

Previdi, S., Faucheur, F., Faucheur, F., Medved, J., and L. Faucheur, "CDNI Footprint Advertisement",

draft-previdi-cdni-footprint-advertisement-01 (work in progress), March 2012.

[I-D.seedorf-alto-for-cdni]

Seedorf, J., "ALTO for CDNi Request Routing",
draft-seedorf-alto-for-cdni-00 (work in progress),
October 2011.

[I-D.vandergaast-edns-client-subnet]

Contavalli, C., Gaast, W., Leach, S., and E. Lewis,
"Client subnet in DNS requests",
draft-vandergaast-edns-client-subnet-01 (work in
progress), April 2012.

[I-D.xiaoyan-cdni-request-routing-protocol]

He, X., Dawkins, S., Li, J., and G. Chen, "Request Routing
Protocol for CDN Interconnection",
draft-xiaoyan-cdni-request-routing-protocol-00 (work in
progress), October 2011.

[RFC3466] Day, M., Cain, B., Tomlinson, G., and P. Rzewski, "A Model
for Content Internetworking (CDI)", RFC 3466,
February 2003.

[RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic
Optimization (ALTO) Problem Statement", RFC 5693,
October 2009.

Authors' Addresses

Larry Peterson (editor)
Verivue, Inc.
2 Technology Park Drive
Westford, MA
USA

Phone: +1 978 303 8032
Email: lpeterson@verivue.com

Bruce Davie
Nicira Networks, Inc.
3460 W. Bayshore Rd.
Palo Alto, CA 94303
USA

Email: bsd@nicira.com

Network Working Group
Internet-Draft
Obsoletes: 3466 (if approved)
Intended status: Informational
Expires: December 8, 2014

L. Peterson
Akamai Technologies, Inc.
B. Davie
VMware, Inc.
R. van Brandenburg, Ed.
TNO
June 6, 2014

Framework for CDN Interconnection
draft-ietf-cdni-framework-14

Abstract

This document presents a framework for Content Distribution Network Interconnection (CDNI). The purpose of the framework is to provide an overall picture of the problem space of CDNI and to describe the relationships among the various components necessary to interconnect CDNs. CDN Interconnection requires the specification of interfaces and mechanisms to address issues such as request routing, distribution metadata exchange, and logging information exchange across CDNs. The intent of this document is to outline what each interface needs to accomplish, and to describe how these interfaces and mechanisms fit together, while leaving their detailed specification to other documents. This document, in combination with RFC 6707, obsoletes RFC 3466.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 8, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Terminology	3
1.2. Reference Model	5
1.3. Structure Of This Document	9
2. Building Blocks	9
2.1. Request Redirection	9
2.1.1. DNS Redirection	9
2.1.2. HTTP Redirection	11
3. Overview of CDNI Operation	11
3.1. Preliminaries	13
3.2. Iterative HTTP Redirect Example	14
3.3. Recursive HTTP Redirection Example	19
3.4. Iterative DNS-based Redirection Example	23
3.4.1. Notes on using DNSSEC	27
3.5. Dynamic Footprint Discovery Example	28
3.6. Content Removal Example	30
3.7. Pre-Positioned Content Acquisition Example	31
3.8. Asynchronous CDNI Metadata Example	32
3.9. Synchronous CDNI Metadata Acquisition Example	34
3.10. Content and Metadata Acquisition with Multiple Upstream CDNs	36
4. Main Interfaces	37
4.1. In-Band versus Out-of-Band Interfaces	38
4.2. Cross Interface Concerns	38
4.3. Request Routing Interfaces	39
4.4. CDNI Logging Interface	40
4.5. CDNI Control Interface	42
4.6. CDNI Metadata Interface	42
4.7. HTTP Adaptive Streaming Concerns	43
4.8. URI Rewriting	44
5. Deployment Models	45

5.1. Meshed CDNs	46
5.2. CSP combined with CDN	47
5.3. CSP using CDNI Request Routing Interface	47
5.4. CDN Federations and CDN Exchanges	48
6. Trust Model	51
7. IANA Considerations	52
8. Privacy Considerations	52
9. Security Considerations	53
9.1. Security of CDNI Interfaces	54
9.2. Digital Rights Management	54
10. Contributors	54
11. Acknowledgements	54
12. Informative References	55
Authors' Addresses	56

1. Introduction

This document provides an overview of the various components necessary to interconnect CDNs, expanding on the problem statement and use cases introduced in [RFC6770] and [RFC6707]. It describes the necessary interfaces and mechanisms in general terms and outlines how they fit together to form a complete system for CDN Interconnection. Detailed specifications are left to other documents. This document makes extensive use of message flow examples to illustrate the operation of interconnected CDNs, but these examples should be considered illustrative rather than prescriptive.

[RFC3466] uses different terminology and models for "Content Internetworking (CDI)". It is also less prescriptive in terms of interfaces. To avoid confusion, this document obsoletes [RFC3466].

1.1. Terminology

This document uses the core terminology defined in [RFC6707]. It also introduces the following terms:

CDN-Domain: a host name (FQDN) at the beginning of a URL (excluding port and scheme), representing a set of content that is served by a given CDN. For example, in the URL `http://cdn.csp.example/...rest of url...`, the CDN domain is `cdn.csp.example`. A major role of CDN-Domain is to identify a region (subset) of the URI space relative to which various CDN Interconnection rules and policies are to apply. For example, a record of CDN Metadata might be defined for the set of resources corresponding to some CDN-Domain.

Distinguished CDN-Domain: a CDN-Domain that is allocated by a CDN for the purposes of communication with a peer CDN, but which is not found

in client requests. Such CDN-Domains may be used for inter-CDN acquisition, or as redirection targets, and enable a CDN to distinguish a request from a peer CDN from an end-user request.

Delivering CDN: the CDN that ultimately delivers a piece of content to the end-user. The last in a potential sequence of downstream CDNs.

Iterative CDNI Request Redirection: When an upstream CDN elects to redirect a request towards a downstream CDN, the upstream CDN can base its redirection purely on a local decision (and without attempting to take into account how the downstream CDN may in turn redirect the user agent). In that case, the upstream CDN redirects the request to the request routing system in the downstream CDN, which in turn will decide how to redirect that request: this approach is referred to as "Iterative" CDNI Request Redirection.

Recursive CDNI Request Redirection: When an upstream CDN elects to redirect a request towards a downstream CDN, the upstream CDN can query the downstream CDN Request Routing system via the CDNI Request Routing Redirection Interface (or use information cached from earlier similar queries) to find out how the downstream CDN wants the request to be redirected. This allows the upstream CDN to factor in the downstream CDN response when redirecting the user agent. This approach is referred to as "Recursive" CDNI Request Redirection. Note that the downstream CDN may elect to have the request redirected directly to a Surrogate inside the downstream CDN, or to any other element in the downstream CDN (or in another CDN) to handle the redirected request appropriately.

Synchronous CDNI operations: operations between CDNs that happen during the process of servicing a user request, i.e. between the time that the user agent begins its attempt to obtain content and the time at which that request is served.

Asynchronous CDNI operations: operations between CDNs that happen independently of any given user request, such as advertisement of footprint information or pre-positioning of content for later delivery.

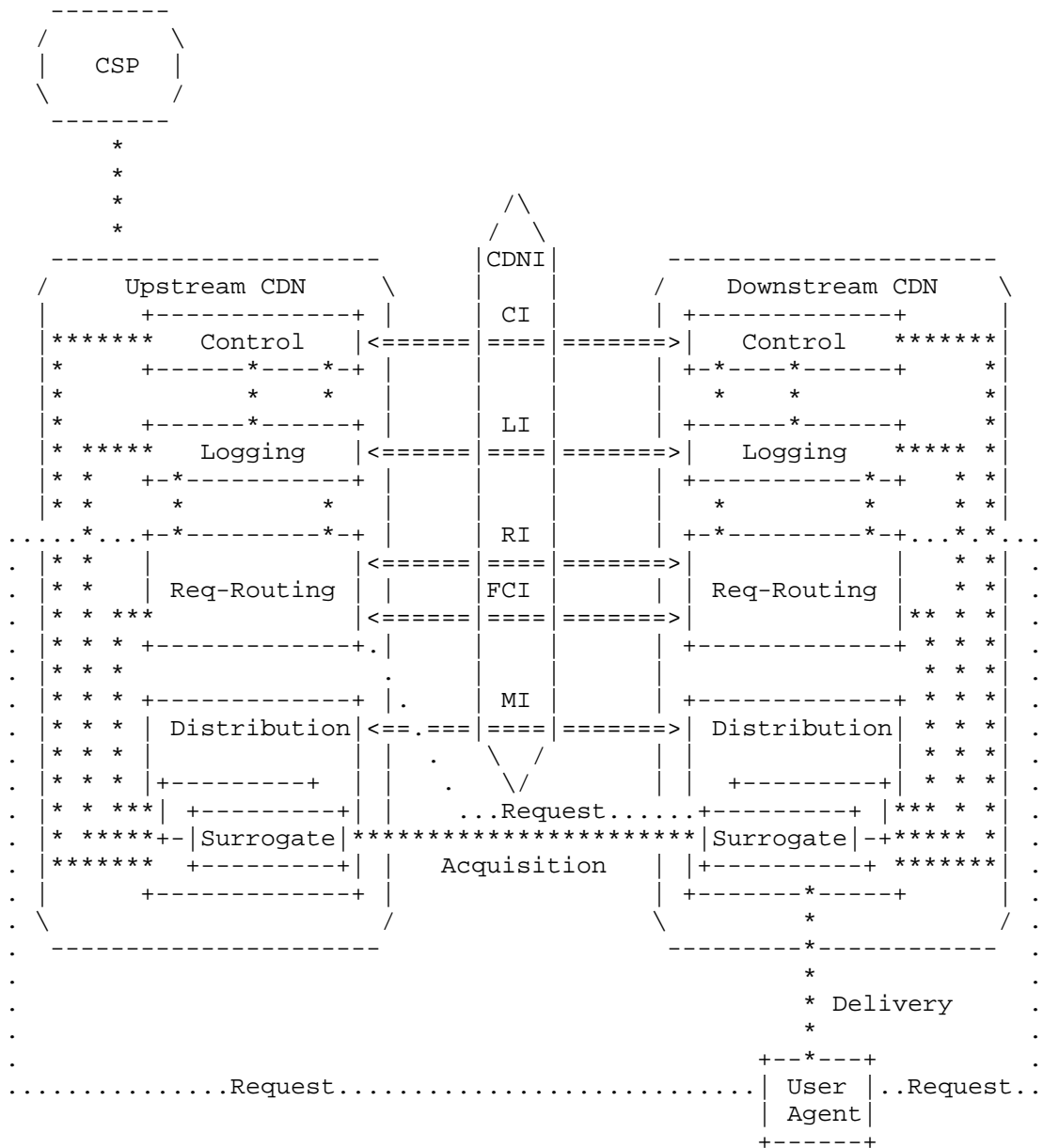
Trigger Interface: a subset of the CDNI Control interface that includes operations to pre-position, revalidate, and purge both metadata and content. These operations are typically called in response to some action (Trigger) by the Content Service Provider (CSP) on the upstream CDN.

We also sometimes use uCDN and dCDN as shorthand for upstream CDN and downstream CDN (see [RFC6707]), respectively.

At various points in this document, the concept of a CDN footprint is used. For a discussion on what constitutes a CDN footprint, the reader is referred to [I-D.ietf-cdni-footprint-capabilities-semantics].

1.2. Reference Model

This document uses the reference model in Figure 1, which expands the reference model originally defined in [RFC6707]. (The difference is that the expanded model splits the Request Routing Interface into its two distinct parts: the Request Routing Redirection interface and the Footprint and Capabilities Advertisement interface, as described below.)



\Leftrightarrow interfaces inside the scope of CDNI

**** and interfaces outside the scope of CDNI

Figure 1: CDNI Expanded Model and CDNI Interfaces

We note that while some interfaces in the reference model are "out of scope" for the CDNI WG (in the sense that there is no need to define new protocols for those interfaces) we still need to refer to them in this document to explain the overall operation of CDNI.

We also note that, while we generally show only one upstream CDN serving a given CSP, it is entirely possible that multiple uCDNs can serve a single CSP. In fact, this situation effectively exists today in the sense that a single CSP can currently delegate its content delivery to more than one CDN.

The following briefly describes the five CDNI interfaces, paraphrasing the definitions given in [RFC6707]. We discuss these interfaces in more detail in Section 4.

- o CDNI Control interface (CI): Operations to bootstrap and parameterize the other CDNI interfaces, as well as operations to pre-position, revalidate, and purge both metadata and content. The latter subset of operations is sometimes collectively called the "Trigger interface."
- o CDNI Request Routing interface: Operations to determine what CDN (and optionally what surrogate within a CDN) is to serve end-user's requests. This interface is actually a logical bundling of two separate but related interfaces:
 - * CDNI Footprint & Capabilities Advertisement interface (FCI): Asynchronous operations to exchange routing information (e.g., the network footprint and capabilities served by a given CDN) that enables CDN selection for subsequent user requests; and
 - * CDNI Request Routing Redirection interface (RI): Synchronous operations to select a delivery CDN (surrogate) for a given user request.
- o CDNI Metadata interface (MI): Operations to communicate metadata that governs how the content is delivered by interconnected CDNs. Examples of CDNI metadata include geo-blocking directives, availability windows, access control mechanisms, and purge directives. It may include a combination of:
 - * Asynchronous operations to exchange metadata that govern subsequent user requests for content; and
 - * Synchronous operations that govern behavior for a given user request for content.

- o CDNI Logging interface (LI): Operations that allow interconnected CDNs to exchange relevant activity logs. It may include a combination of:
 - * Real-time exchanges, suitable for runtime traffic monitoring; and
 - * Offline exchanges, suitable for analytics and billing.

The division between the sets of Trigger-based operations in the CDNI Control interface and the CDNI Metadata interface is somewhat arbitrary. For both cases, the information passed from the upstream CDN to the downstream CDN can broadly be viewed as metadata that describes how content is to be managed by the downstream CDN. For example, the information conveyed by CI to pre-position, revalidate or purge metadata is similar to the information conveyed by posting updated metadata via the MI. Even the CI operation to purge content could be viewed as a metadata update for that content: purge simply says that the availability window for the named content ends now. The two interfaces share much in common, so minimally, there will need to be a consistent data model that spans both.

The distinction we draw has to do with what the uCDN knows about the successful application of the metadata by the dCDN. In the case of the CI, the downstream CDN returning a successful status message guarantees that the operation has been successfully completed; e.g., the content has been purged or pre-positioned. This implies that the downstream CDN accepts responsibility for having successfully completed the requested operation. In contrast, metadata passed between CDNs via the MI carries no such completion guarantee. Returning success implies successful receipt of the metadata, but nothing can be inferred about precisely when the metadata will take effect in the downstream CDN, only that it will take effect eventually. This is because of the challenge in globally synchronizing updates to metadata with end-user requests that are currently in progress (or indistinguishable from currently being in progress). Clearly, a CDN will not be viewed as a trusted peer if "eventually" often becomes an indefinite period of time, but the acceptance of responsibility cannot be as crisply defined for the MI.

Finally, there is a practical issue that impacts all of the CDNI interfaces, and that is whether or not to optimize CDNI for HTTP Adaptive Streaming (HAS). We highlight specific issues related to delivering HAS content throughout this document, but for a more thorough treatment of the topic, see [RFC6983].

1.3. Structure Of This Document

The remainder of this document is organized as follows:

- o Section 2 describes some essential building blocks for CDNI, notably the various options for redirecting user requests to a given CDN.
- o Section 3 provides a number of illustrative examples of various CDNI operations.
- o Section 4 describes the functionality of the main CDNI interfaces.
- o Section 5 shows how various deployment models of CDNI may be achieved using the defined interfaces.
- o Section 6 describes the trust model of CDNI and the issues of transitive trust in particular that CDNI raises.

2. Building Blocks

2.1. Request Redirection

At its core, CDN Interconnection requires the redirection of requests from one CDN to another. For any given request that is received by an upstream CDN, it will either respond to the request directly, or somehow redirect the request to a downstream CDN. Two main mechanisms are available for redirecting a request to a downstream CDN. The first leverages the DNS name resolution process and the second uses application-layer redirection mechanisms such as the HTTP 302 or RTSP 302 redirection responses. While there exists a large variety of application-layer protocols that include some form of redirection mechanism, this document will use HTTP (and HTTPS) in its examples. Similar mechanisms can be applied to other application-layer protocols. What follows is a short discussion of both DNS- and HTTP-based redirection, before presenting some examples of their use in Section 3.

2.1.1. DNS Redirection

DNS redirection is based on returning different IP addresses for the same DNS name, for example, to balance server load or to account for the client's location in the network. A DNS server, sometimes called the Local DNS (LDNS), resolves DNS names on behalf of an end-user. The LDNS server in turn queries other DNS servers until it reaches the authoritative DNS server for the CDN-Domain. The network operator typically provides the LDNS server, although the user is free to choose other DNS servers (e.g., OpenDNS, Google Public DNS).

This latter possibility is important because the authoritative DNS server sees only the IP address of the DNS server that queries it, not the IP address of the original end-user.

The advantage of DNS redirection is that it is completely transparent to the end user; the user sends a DNS name to the LDNS server and gets back an IP address. On the other hand, DNS redirection is problematic because the DNS request comes from the LDNS server, not the end-user. This may affect the accuracy of server selection that is based on the user's location. The transparency of DNS redirection is also a problem in that there is no opportunity to take the attributes of the user agent or the URI path component into account. We consider two main forms of DNS redirection: simple and CNAME-based.

In simple DNS redirection, the authoritative DNS server for the name simply returns an IP address from a set of possible IP addresses. The answer is chosen from the set based on characteristics of the set (e.g., the relative loads on the servers) or characteristics of the client (e.g., the location of the client relative to the servers). Simple redirection is straightforward. The only caveats are (1) there is a limit to the number of alternate IP addresses a single DNS server can manage; and (2) DNS responses are cached by downstream servers so the TTL on the response must be set to an appropriate value so as to preserve the freshness of the redirection.

In CNAME-based DNS redirection, the authoritative server returns a CNAME response to the DNS request, telling the LDNS server to restart the name lookup using a new name. A CNAME is essentially a symbolic link in the DNS namespace, and like a symbolic link, redirection is transparent to the client; the LDNS server gets the CNAME response and re-executes the lookup. Only when the name has been resolved to an IP address does it return the result to the user. Note that DNAME would be preferable to CNAME if it becomes widely supported.

One of the advantages of DNS redirection compared to HTTP redirection is that it can be cached, reducing load on the redirecting CDN's DNS server. However, this advantage can also be a drawback, especially when a given DNS resolver doesn't strictly adhere to the TTL, which is a known problem in some real world environments. In such cases, an end-user might end up at a dCDN without first having passed through the uCDN, which might be an undesirable scenario from a uCDN point of view.

2.1.2. HTTP Redirection

HTTP redirection makes use of the redirection response of the HTTP protocol (e.g., "302" or "307"). This response contains a new URL that the application should fetch instead of the original URL. By changing the URL appropriately, the server can cause the user to redirect to a different server. The advantages of HTTP redirection are that (1) the server can change the URL fetched by the client to include, for example, both the DNS name of the particular server to use, as well as the original HTTP server that was being accessed; (2) the client sends the HTTP request to the server, so that its IP address is known and can be used in selecting the server; and (3) other attributes (e.g., content type, user agent type) are visible to the redirection mechanism.

Just as is the case for DNS redirection, there are some potential disadvantages of using HTTP redirection. For example, it may affect application behavior, e.g. web browsers will not send cookies if the URL changes to a different domain. In addition, although this might also be an advantage, results of HTTP redirection are not cached so that all redirections must go through to the uCDN.

3. Overview of CDNI Operation

To provide a big picture overview of the various components of CDN Interconnection, we walk through a "day in the life" of a content item that is made available via a pair of interconnected CDNs. This will serve to illustrate many of the functions that need to be supported in a complete CDNI solution. We give examples using both DNS-based and HTTP-based redirection. We begin with very simple examples and then show how additional capabilities, such as recursive request redirection and content removal, might be added.

Before walking through the specific examples, we present a high-level view of the operations that may take place. This high-level overview is illustrated in Figure 2. Note that most operations will involve only a subset of all the messages shown below, and that the order and number of operations may vary considerably, as the more detailed examples illustrate.

The following shows Operator A as the upstream CDN (uCDN) and Operator B as the downstream CDN (dCDN), where the former has a relationship with a content provider and the latter being the CDN selected by Operator A to deliver content to the end-user. The interconnection relationship may be symmetric between these two CDN operators, but each direction can be considered as operating independently of the other so for simplicity we show the interaction in one direction only.

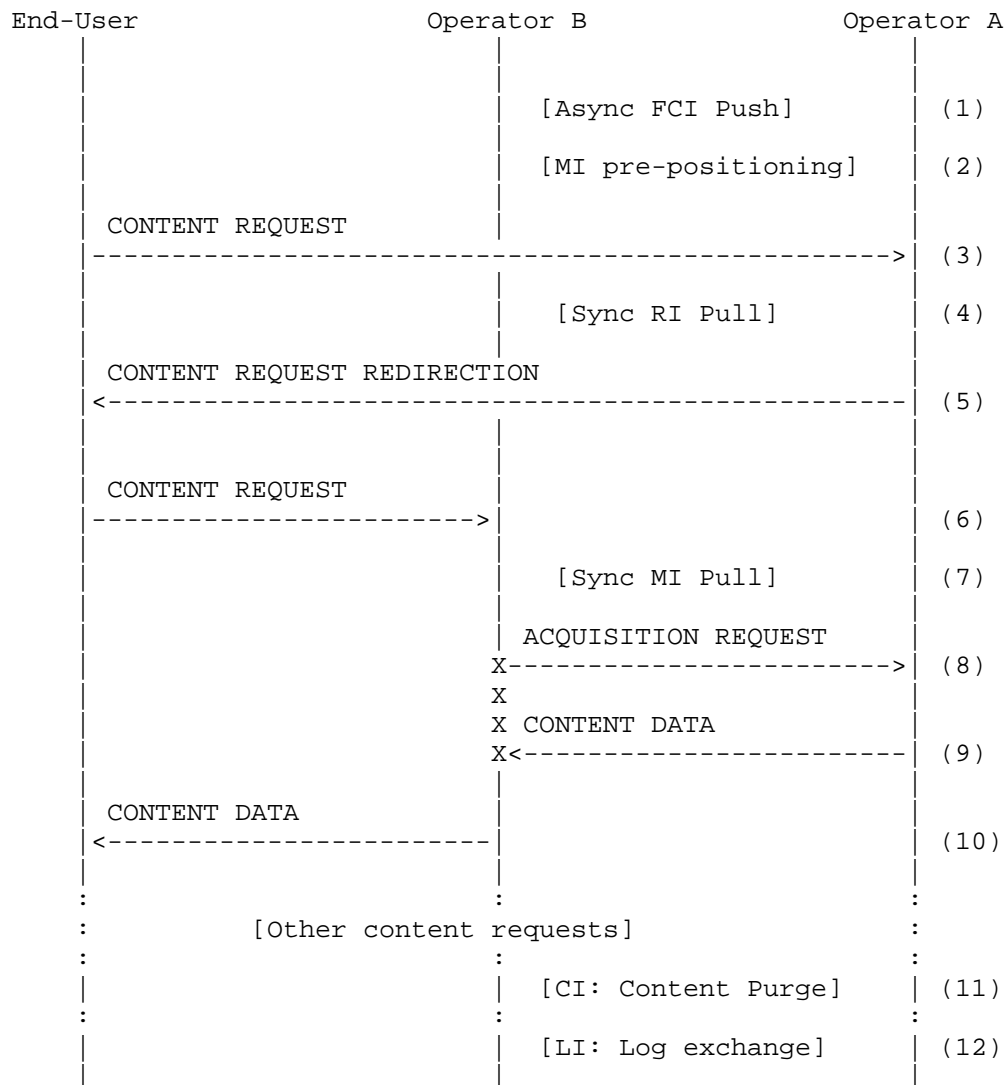


Figure 2: Overview of Operation

The operations shown in the Figure are as follows:

1. dCDN uses the FCI to advertise information relevant to its delivery footprint and capabilities prior to any content requests being redirected.

2. Prior to any content request, the uCDN uses the MI to pre-position CDNI metadata to the dCDN, thereby making that metadata available in readiness for later content requests.
3. A content request from a user agent arrives at uCDN.
4. uCDN may use the RI to synchronously request information from dCDN regarding its delivery capabilities to decide if dCDN is a suitable target for redirection of this request.
5. uCDN redirects the request to dCDN by sending some response (DNS, HTTP) to the user agent.
6. The user agent requests the content from dCDN.
7. dCDN may use the MI to synchronously request metadata related to this content from uCDN, e.g. to decide whether to serve it.
8. If the content is not already in a suitable cache in dCDN, dCDN may acquire it from uCDN.
9. The content is delivered to dCDN from uCDN.
10. The content is delivered to the user agent by dCDN.
11. Some time later, perhaps at the request of the CSP (not shown) uCDN may use the CI to instruct dCDN to purge the content, thereby ensuring it is not delivered again.
12. After one or more content delivery actions by dCDN, a log of delivery actions may be provided to uCDN using the LI.

The following sections show some more specific examples of how these operations may be combined to perform various delivery, control and logging operations across a pair of CDNs.

3.1. Preliminaries

Initially, we assume that there is at least one CSP that has contracted with an upstream CDN (uCDN) to deliver content on its behalf. We are not particularly concerned with the interface between the CSP and uCDN, other than to note that it is expected to be the same as in the "traditional" (non-interconnected) CDN case. Existing mechanisms such as DNS CNAMEs or HTTP redirects (Section 2) can be used to direct a user request for a piece of content from the CSP towards the CSP's chosen upstream CDN.

We assume Operator A provides an upstream CDN that serves content on behalf of a CSP with CDN-Domain `cdn.csp.example`. We assume that Operator B provides a downstream CDN. An end user at some point makes a request for URL

`http://cdn.csp.example/...rest of url...`

It may well be the case that `cdn.csp.example` is just a CNAME for some other CDN-Domain (such as `csp.op-a.example`). Nevertheless, the HTTP request in the examples that follow is assumed to be for the example URL above.

Our goal is to enable content identified by the above URL to be served by the CDN of operator B. In the following sections we will walk through some scenarios in which content is served, as well as other CDNI operations such as the removal of content from a downstream CDN.

3.2. Iterative HTTP Redirect Example

In this section we walk through a simple, illustrative example using HTTP redirection from uCDN to dCDN. The example also assumes the use of HTTP redirection inside uCDN and dCDN; however, this is independent of the choice of redirection approach across CDNs, so an alternative example could be constructed still showing HTTP redirection from uCDN to dCDN but using DNS for handling of request inside each CDN.

We assume for this example that Operators A and B have established an agreement to interconnect their CDNs, with A being upstream and B being downstream.

The operators agree that a CDN-Domain `peer-a.op-b.example` will be used as the target of redirections from uCDN to dCDN. We assume the name of this domain is communicated by some means to each CDN. (This could be established out-of-band or via a CDNI interface.) We refer to this domain as a "distinguished" CDN-Domain to convey the fact that its use is limited to the interconnection mechanism; such a domain is never used directly by a CSP.

We assume the operators also agree on some distinguished CDN-Domain that will be used for inter-CDN acquisition of CSP's content from uCDN by dCDN. In this example, we'll use `op-b-acq.op-a.example`.

We assume the operators also exchange information regarding which requests dCDN is prepared to serve. For example, dCDN may be prepared to serve requests from clients in a given geographical

region or a set of IP address prefixes. This information may again be provided out of band or via a defined CDNI interface.

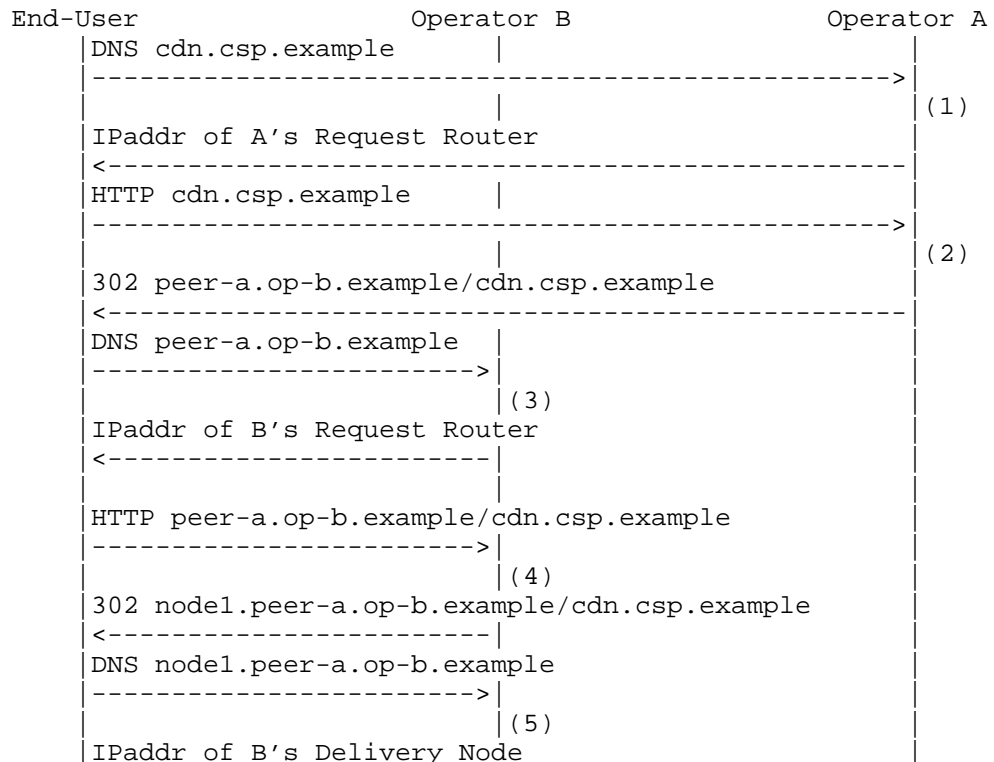
We assume DNS is configured in the following way:

- o The content provider is configured to make operator A the authoritative DNS server for `cdn.csp.example` (or to return a CNAME for `cdn.csp.example` for which operator A is the authoritative DNS server).
- o Operator A is configured so that a DNS request for `op-b-acq.op-a.example` returns a request router in Operator A.
- o Operator B is configured so that a DNS request for `peer-a.op-b.example/cdn.csp.example` returns a request router in Operator B.

Figure 3 illustrates how a client request for

`http://cdn.csp.example/...rest of url...`

is handled.



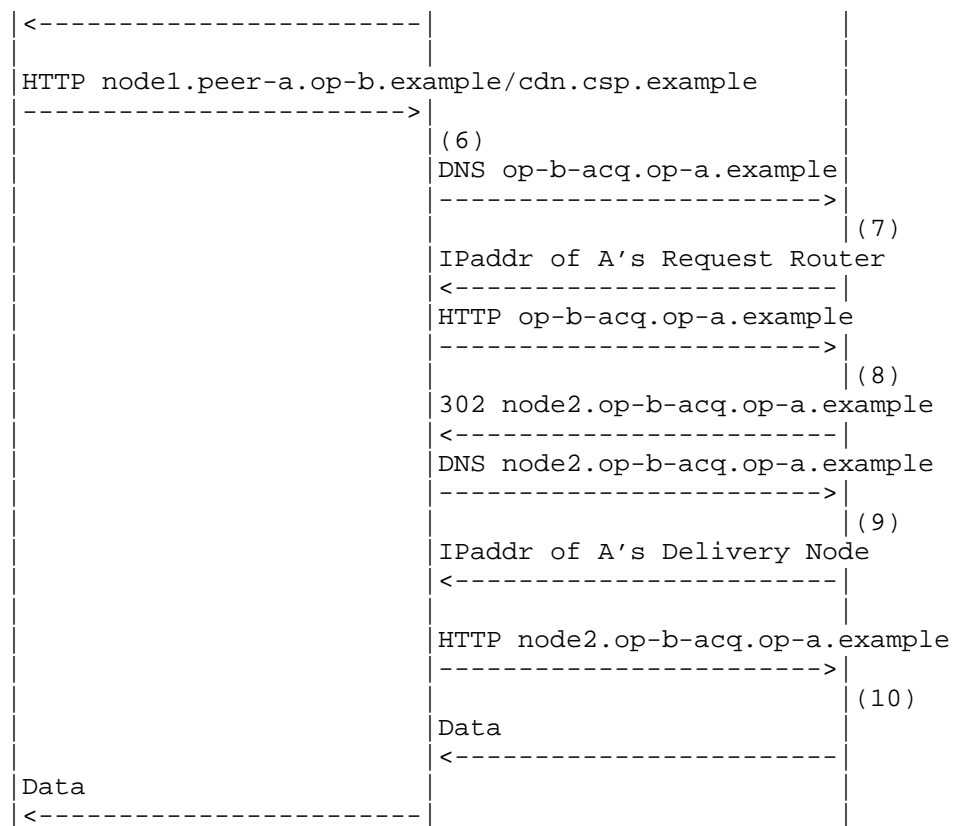


Figure 3: Message Flow for Iterative HTTP Redirection

The steps illustrated in the figure are as follows:

1. A DNS resolver for Operator A processes the DNS request for its customer based on CDN-Domain `cdn.csp.example`. It returns the IP address of a request router in Operator A.
2. A Request Router for Operator A processes the HTTP request and recognizes that the end-user is best served by another CDN, specifically one provided by Operator B, and so it returns a 302 redirect message for a new URL constructed by "stacking" Operator B's distinguished CDN-Domain (`peer-a.op-b.example`) on the front of the original URL. (Note that more complex URL manipulations are possible, such as replacing the initial CDN-Domain by some opaque handle.)
3. The end-user does a DNS lookup using Operator B's distinguished CDN-Domain (`peer-a.op-b.example`). B's DNS resolver returns the

IP address of a request router for Operator B. Note that if request routing within dCDN was performed using DNS instead of HTTP redirection, B's DNS resolver would also behave as the request router and directly return the IP address of a delivery node.

4. The request router for Operator B processes the HTTP request and selects a suitable delivery node to serve the end-user request, and returns a 302 redirect message for a new URL constructed by replacing the hostname with a subdomain of the Operator B's distinguished CDN-Domain that points to the selected delivery node.
5. The end-user does a DNS lookup using Operator B's delivery node subdomain (node1.peer-a.op-b.example). B's DNS resolver returns the IP address of the delivery node.
6. The end-user requests the content from B's delivery node. In the case of a cache hit, steps 6, 7, 8, 9 and 10 below do not happen, and the content data is directly returned by the delivery node to the end-user. In the case of a cache miss, the content needs to be acquired by dCDN from uCDN (not the CSP). The distinguished CDN-Domain peer-a.op-b.example indicates to dCDN that this content is to be acquired from uCDN; stripping the CDN-Domain reveals the original CDN-Domain cdn.csp.example and dCDN may verify that this CDN-Domain belongs to a known peer (so as to avoid being tricked into serving as an open proxy). It then does a DNS request for an inter-CDN acquisition CDN-Domain as agreed above (in this case, op-b-acq.op-a.example).
7. Operator A's DNS resolver processes the DNS request and returns the IP address of a request router in operator A.
8. The request router for Operator A processes the HTTP request from Operator B delivery node. Operator A request router recognizes that the request is from a peer CDN rather than an end-user because of the dedicated inter-CDN acquisition domain (op-b-acq.op-a.example). (Note that without this specially defined inter-CDN acquisition domain, operator A would be at risk of redirecting the request back to operator B, resulting in an infinite loop). The request router for Operator A selects a suitable delivery node in uCDN to serve the inter-CDN acquisition request and returns a 302 redirect message for a new URL constructed by replacing the hostname with a subdomain of the Operator A's distinguished inter-CDN acquisition domain that points to the selected delivery node.

9. Operator A DNS resolver processes the DNS request and returns the IP address of the delivery node in operator A.
10. Operator B requests (acquires) the content from Operator A. Although not shown, Operator A processes the rest of the URL: it extracts information identifying the origin server, validates that this server has been registered, and determines the content provider that owns the origin server. It may also perform its own content acquisition steps if needed before returning the content to dCDN.

The main advantage of this design is that it is simple: each CDN need only know the distinguished CDN-Domain for each peer, with the upstream CDN "pushing" the downstream CDN-Domain onto the URL as part of its redirect (step 2) and the downstream CDN "popping" its CDN-Domain off the URL to expose a CDN-Domain that the upstream CDN can correctly process. Neither CDN needs to be aware of the internal structure of the other's URLs. Moreover, the inter-CDN redirection is entirely supported by a single HTTP redirect; neither CDN needs to be aware of the other's internal redirection mechanism (i.e., whether it is DNS or HTTP based).

One disadvantage is that the end-user's browser is redirected to a new URL that is not in the same domain of the original URL. This has implications on a number of security or validation mechanisms sometimes used on endpoints. For example, it is important that any redirected URL be in the same domain (e.g., csp.example) if the browser is expected to send any cookies associated with that domain. As another example, some video players enforce validation of a cross domain policy that needs to accommodate the domains involved in the CDN redirection. These problems are generally solvable, but the solutions complicate the example, so we do not discuss them further in this document.

We note that this example begins to illustrate some of the interfaces that may be required for CDNI, but does not require all of them. For example, obtaining information from dCDN regarding the set of client IP addresses or geographic regions it might be able to serve is an aspect of request routing (specifically of the CDNI Footprint & Capabilities Advertisement interface). Important configuration information such as the distinguished names used for redirection and inter-CDN acquisition could also be conveyed via a CDNI interface (e.g., perhaps the CDNI Control interface). The example also shows how existing HTTP-based methods suffice for the acquisition interface. Arguably, the absolute minimum metadata required for CDNI is the information required to acquire the content, and this information was provided "in-band" in this example by means of the URI handed to the client in the HTTP 302 response. The example also

assumes that the CSP does not require any distribution policy (e.g. time window, geo-blocking) or delivery processing to be applied by the interconnected CDNs. Hence, there is no explicit CDNI Metadata interface invoked in this example. There is also no explicit CDNI Logging interface discussed in this example.

We also note that the step of deciding when a request should be redirected to dCDN rather than served by uCDN has been somewhat glossed over. It may be as simple as checking the client IP address against a list of prefixes, or it may be considerably more complex, involving a wide range of factors, such as the geographic location of the client (perhaps determined from a third party service), CDN load, or specific business rules.

This example uses the "iterative" CDNI request redirection approach. That is, uCDN performs part of the request redirection function by redirecting the client to a request router in the dCDN, which then performs the rest of the redirection function by redirecting to a suitable surrogate. If request routing is performed in the dCDN using HTTP redirection, this translates in the end-user experiencing two successive HTTP redirections. By contrast, the alternative approach of "recursive" CDNI request redirection effectively coalesces these two successive HTTP redirections into a single one, sending the end-user directly to the right delivery node in the dCDN. This "recursive" CDNI request routing approach is discussed in the next section.

While the example above uses HTTP, the iterative HTTP redirection mechanism would work over HTTPS in a similar fashion. In order to make sure an end-user's HTTPS request is not downgraded to HTTP along the redirection path, it is necessary for every request router along the path from the initial uCDN Request Router to the final surrogate in the dCDN to respond to an incoming HTTPS request with an HTTP Redirect containing an HTTPS URL. It should be noted that using HTTPS will have the effect of increasing the total redirection process time and increasing the load on the request routers, especially when the redirection path includes many redirects and thus many TLS/SSL sessions. In such cases, a recursive HTTP redirection mechanism, as described in an example in the next section, might help to reduce some of these issues.

3.3. Recursive HTTP Redirection Example

The following example builds on the previous one to illustrate the use of the request routing interface (specifically the CDNI Request Routing Redirection interface) to enable "recursive" CDNI request routing. We build on the HTTP-based redirection approach because it illustrates the principles and benefits clearly, but it is equally

possible to perform recursive redirection when DNS-based redirection is employed.

In contrast to the prior example, the operators need not agree in advance on a CDN-Domain to serve as the target of redirections from uCDN to dCDN. We assume that the operators agree on some distinguished CDN-Domain that will be used for inter-CDN acquisition of CSP's content by dCDN. In this example, we'll use op-b-acq.op-a.example.

We assume the operators also exchange information regarding which requests dCDN is prepared to serve. For example, dCDN may be prepared to serve requests from clients in a given geographical region or a set of IP address prefixes. This information may again be provided out of band or via a defined protocol.

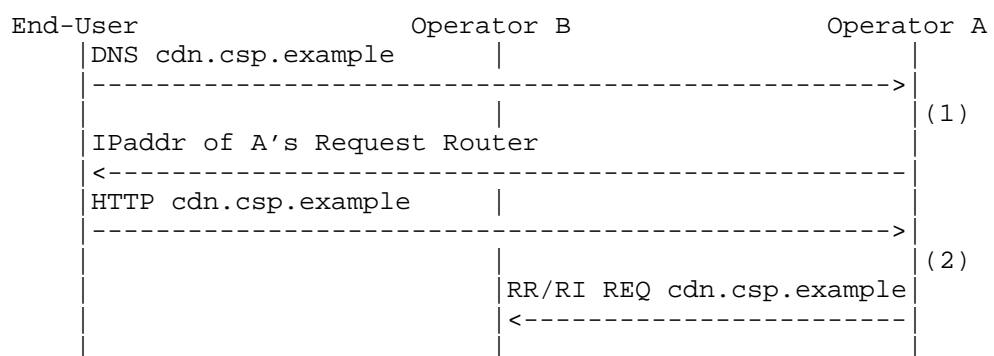
We assume DNS is configured in the following way:

- o The content provider is configured to make operator A the authoritative DNS server for cdn.csp.example (or to return a CNAME for cdn.csp.example for which operator A is the authoritative DNS server).
- o Operator A is configured so that a DNS request for op-b-acq.op-a.example returns a request router in Operator A.
- o Operator B is configured so that a request for node1.op-b.example/cdn.csp.example returns the IP address of a delivery node. Note that there might be a number of such delivery nodes.

Figure 3 illustrates how a client request for

http://cdn.csp.example/...rest of url...

is handled.



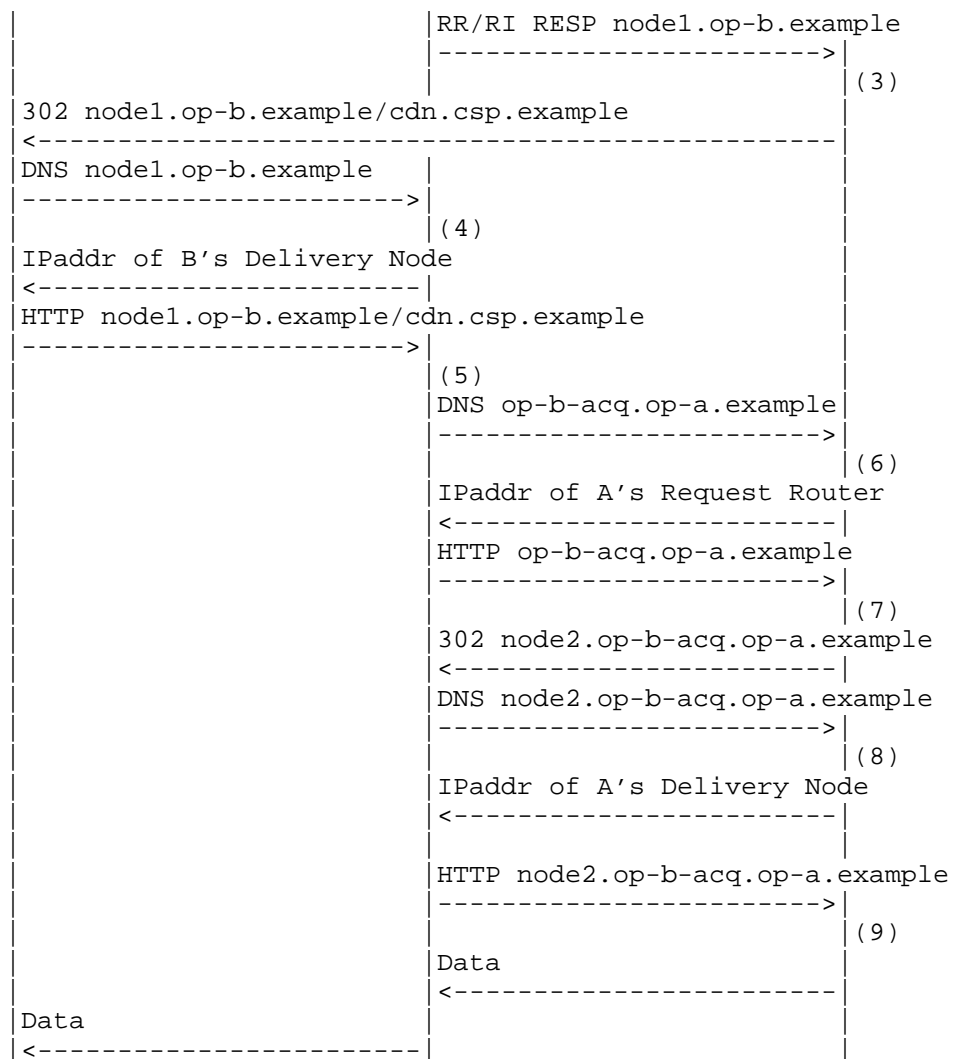


Figure 4: Message Flow for Recursive HTTP Redirection

The steps illustrated in the figure are as follows:

1. A DNS resolver for Operator A processes the DNS request for its customer based on CDN-Domain `cdn.csp.example`. It returns the IP address of a Request Router in Operator A.
2. A Request Router for Operator A processes the HTTP request and recognizes that the end-user is best served by another CDN--specifically one provided by Operator B--and so it queries the

CDNI Request Routing Redirection interface of Operator B, providing a set of information about the request including the URL requested. Operator B replies with the DNS name of a delivery node.

3. Operator A returns a 302 redirect message for a new URL obtained from the RI.
4. The end-user does a DNS lookup using the host name of the URL just provided (node1.op-b.example). B's DNS resolver returns the IP address of the corresponding delivery node. Note that, since the name of the delivery node was already obtained from B using the RI, there should not be any further redirection here (in contrast to the iterative method described above.)
5. The end-user requests the content from B's delivery node, potentially resulting in a cache miss. In the case of a cache miss, the content needs to be acquired from uCDN (not the CSP.) The distinguished CDN-Domain op-b.example indicates to dCDN that this content is to be acquired from another CDN; stripping the CDN-Domain reveals the original CDN-Domain cdn.csp.example, dCDN may verify that this CDN-Domain belongs to a known peer (so as to avoid being tricked into serving as an open proxy). It then does a DNS request for the inter-CDN Acquisition "distinguished" CDN-Domain as agreed above (in this case, op-b-acq.op-a.example).
6. Operator A DNS resolver processes the DNS request and returns the IP address of a request router in operator A.
7. The request router for Operator A processes the HTTP request from Operator B delivery node. Operator A request router recognizes that the request is from a peer CDN rather than an end-user because of the dedicated inter-CDN acquisition domain (op-b-acq.op-a.example). (Note that without this specially defined inter-CDN acquisition domain, operator A would be at risk of redirecting the request back to operator B, resulting in an infinite loop). The request router for Operator A selects a suitable delivery node in uCDN to serve the inter-CDN acquisition request and returns a 302 redirect message for a new URL constructed by replacing the hostname with a subdomain of the Operator A's distinguished inter-CDN acquisition domain that points to the selected delivery node.
8. Operator A recognizes that the DNS request is from a peer CDN rather than an end-user (due to the internal CDN-Domain) and so returns the address of a delivery node. (Note that without this specially defined internal domain, Operator A would be at risk of

redirecting the request back to Operator B, resulting in an infinite loop.)

9. Operator B requests (acquires) the content from Operator A. Operator A serves content for the requested CDN-Domain to dCDN. Although not shown, it is at this point that Operator A processes the rest of the URL: it extracts information identifying the origin server, validates that this server has been registered, and determines the content provider that owns the origin server. It may also perform its own content acquisition steps if needed before returning the content to dCDN.

Recursive redirection has the advantage over iterative of being more transparent from the end-user's perspective, but the disadvantage of each CDN exposing more of its internal structure (in particular, the addresses of edge caches) to peer CDNs. By contrast, iterative redirection does not require dCDN to expose the addresses of its edge caches to uCDN.

This example happens to use HTTP-based redirection in both CDN A and CDN B, but a similar example could be constructed using DNS-based redirection in either CDN. Hence, the key point to take away here is simply that the end user only sees a single redirection of some type, as opposed to the pair of redirections in the prior (iterative) example.

The use of the RI requires that the request routing mechanism be appropriately configured and bootstrapped, which is not shown here. More discussion on the bootstrapping of interfaces is provided in Section 4

3.4. Iterative DNS-based Redirection Example

In this section we walk through a simple example using DNS-based redirection for request redirection from uCDN to dCDN (as well as for request routing inside dCDN and uCDN). As noted in Section 2.1, DNS-based redirection has certain advantages over HTTP-based redirection (notably, it is transparent to the end-user) as well as some drawbacks (notably the client IP address is not visible to the request router).

As before, Operator A has to learn the set of requests that dCDN is willing or able to serve (e.g. which client IP address prefixes or geographic regions are part of the dCDN footprint). We assume Operator B has and makes known to Operator A some unique identifier that can be used for the construction of a distinguished CDN-Domain, as shown in more detail below. (This identifier strictly needs only to be unique within the scope of Operator A, but a globally unique

identifier, such as an AS number assigned to B, is one easy way to achieve that.) Also, Operator A obtains the NS records for Operator B's externally visible redirection servers. Also, as before, a distinguished CDN-Domain, such as `op-b-acq.op-a.example`, must be assigned for inter-CDN acquisition.

We assume DNS is configured in the following way:

- o The CSP is configured to make Operator A the authoritative DNS server for `cdn.csp.example` (or to return a CNAME for `cdn.csp.example` for which operator A is the authoritative DNS server).
- o When uCDN sees a request best served by dCDN, it returns CNAME and NS records for `"b.cdn.csp.example"`, where "b" is the unique identifier assigned to Operator B. (It may, for example, be an AS number assigned to Operator B.)
- o dCDN is configured so that a request for `"b.cdn.csp.example"` returns a delivery node in dCDN.
- o uCDN is configured so that a request for `"op-b-acq.op-a.example"` returns a delivery node in uCDN.

Figure 5 depicts the exchange of DNS and HTTP requests. The main differences from Figure 3 are the lack of HTTP redirection and transparency to the end-user.

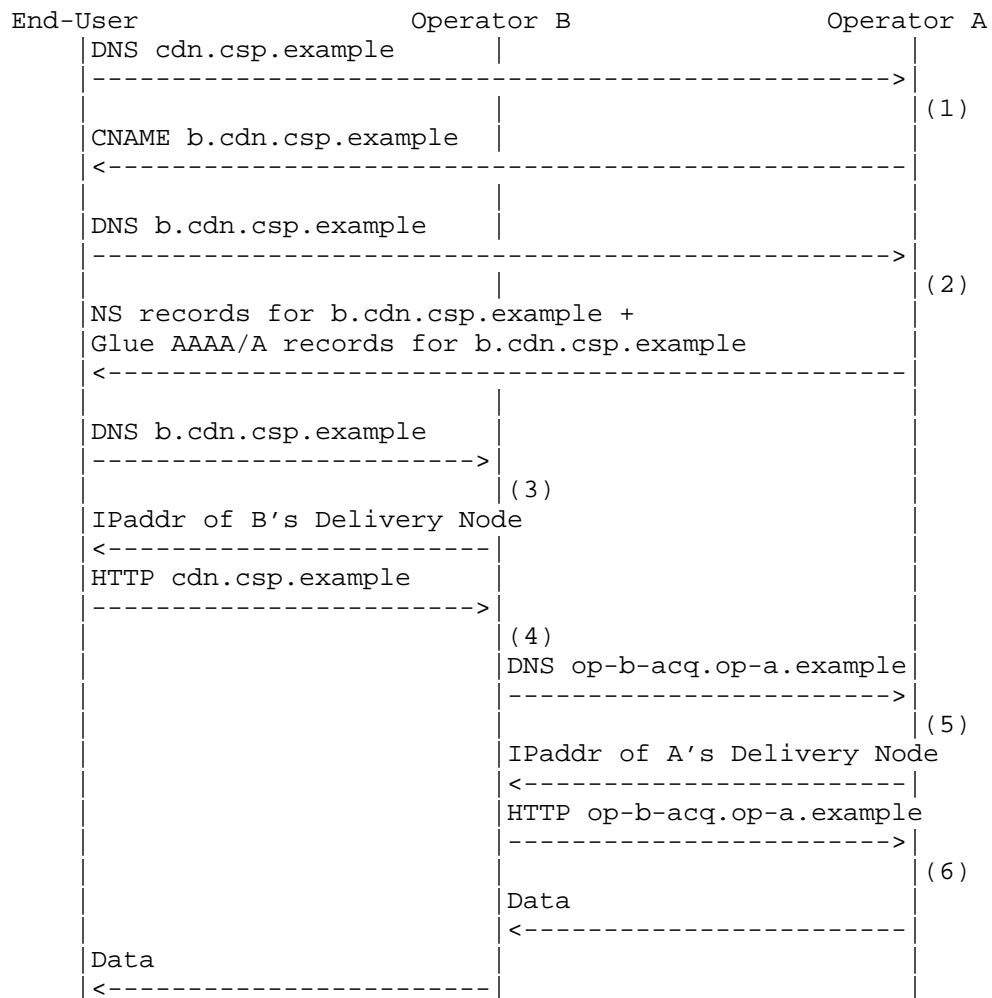


Figure 5: Message Flow for DNS-based Redirection

The steps illustrated in the figure are as follows:

1. Request Router for Operator A processes the DNS request for CDN-Domain `cdn.csp.example` and recognizes that the end-user is best served by another CDN. (This may depend on the IP address of the user's local DNS resolver, or other information discussed below.) The Request Router returns a DNS CNAME response by "stacking" the distinguished identifier for Operator B onto the original CDN-Domain (e.g., `b.cdn.csp.example`).

2. The end-user sends a DNS query for the modified CDN-Domain (i.e. b.cdn.csp.example) to Operator A's DNS server. The Request Router for Operator A processes the DNS request and return a delegation to b.cdn.csp.example by sending an NS record plus glue AAAA/A records pointing to Operator B's DNS server. (This extra step is necessary since typical DNS implementation won't follow an NS record when it is sent together with a CNAME record, thereby necessitating a two-step approach).
3. The end-user sends a DNS query for the modified CDN-Domain (i.e., b.cdn.csp.example) to Operator B's DNS server, using the NS and AAAA/A records received in step 2. This causes B's Request Router to respond with a suitable delivery node.
4. The end-user requests the content from B's delivery node. The requested URL contains the name cdn.csp.example. (Note that the returned CNAME does not affect the URL.) At this point the delivery node has the correct IP address of the end-user and can do an HTTP 302 redirect if the redirections in steps 2 and 3 were incorrect. Otherwise B verifies that this CDN-Domain belongs to a known peer (so as to avoid being tricked into serving as an open proxy). It then does a DNS request for an "internal" CDN-Domain as agreed above (op-b-acq.op-a.example).
5. Operator A recognizes that the DNS request is from a peer CDN rather than an end-user (due to the internal CDN-Domain) and so returns the address of a delivery node in uCDN.
6. Operator A serves content to dCDN. Although not shown, it is at this point that Operator A processes the rest of the URL: it extracts information identifying the origin server, validates that this server has been registered, and determines the content provider that owns the origin server.

The advantages of this approach are that it is more transparent to the end-user and requires fewer round trips than HTTP-based redirection (in its worst case, i.e., when none of the needed DNS information is cached). A potential problem is that the upstream CDN depends on being able to learn the correct downstream CDN that serves the end-user from the client address in the DNS request. In standard DNS operation, uCDN will only obtain the address of the client's local DNS resolver (LDNS), which is not guaranteed to be in the same network (or geographic region) as the client. If not--e.g., the end-user uses a global DNS service--then the upstream CDN cannot determine the appropriate downstream CDN to serve the end-user. In this case, and assuming the uCDN is capable of detecting that situation, one option is for the upstream CDN to treat the end-user as it would any user not connected to a peer CDN. Another option is

for the upstream CDN to "fall back" to a pure HTTP-based redirection strategy in this case (i.e., use the first method). Note that this problem affects existing CDNs that rely on DNS to determine where to redirect client requests, but the consequences are arguably less serious for CDNI since the LDNS is likely in the same network as the dCDN serves.

As with the prior example, this example partially illustrates the various interfaces involved in CDNI. Operator A could learn dynamically from Operator B the set of prefixes or regions that B is willing and able to serve via the CDNI Footprint & Capabilities Advertisement interface. The distinguished name used for acquisition and the identifier for Operator B that is prepended to the CDN-Domain on redirection are examples of information elements that might also be conveyed by CDNI interfaces (or, alternatively, statically configured). As before, minimal metadata sufficient to obtain the content is carried "in-band" as part of the redirection process, and standard HTTP is used for inter-CDN acquisition. There is no explicit CDNI Logging interface discussed in this example.

3.4.1. Notes on using DNSSEC

Although it is possible to use DNSSEC in combination with the Iterative DNS-based Redirection mechanism explained above, it is important to note that the uCDN might have to sign records on the fly, since the CNAME returned, and thus the signature provided, can potentially be different for each incoming query. Although there is nothing preventing a uCDN from performing such on-the-fly signing, this might be computationally expensive. In the case where the number of dCDNs, and thus the number of different CNAMEs to return, is relatively stable, an alternative solution would be for the uCDN to pre-generate signatures for all possible CNAMEs. For each incoming query the uCDN would then determine the appropriate CNAME and return it together with the associated pre-generated signature. Note: In the latter case maintaining the serial and signature of SOA might be an issue since technically it should change every time a different CNAME is used. However, since in practice direct SOA queries are relatively rare, a uCDN could defer incrementing the serial and resigning the SOA until it is queried and then do it on-the-fly.

Note also that the NS record and the glue AAAA/A records used in step 2 in the previous section should generally be identical to those of their authoritative zone managed by Operator B. Even if they differ, this will not make the DNS resolution process fail, but the client DNS server will prefer the authoritative data in its cache and use it for subsequent queries. Such inconsistency is a general operational issue of DNS, but it may be more important for this architecture

because the uCDN (operator A) would rely on the consistency to make the resulting redirection work as intended. In general, it is the administrator's responsibility to make them consistent.

3.5. Dynamic Footprint Discovery Example

There could be situations where being able to dynamically discover the set of requests that a given dCDN is willing and able to serve is beneficial. For example, a CDN might at one time be able to serve a certain set of client IP prefixes, but that set might change over time due to changes in the topology and routing policies of the IP network. The following example illustrates this capability. We have chosen the example of DNS-based redirection, but HTTP-based redirection could equally well use this approach.

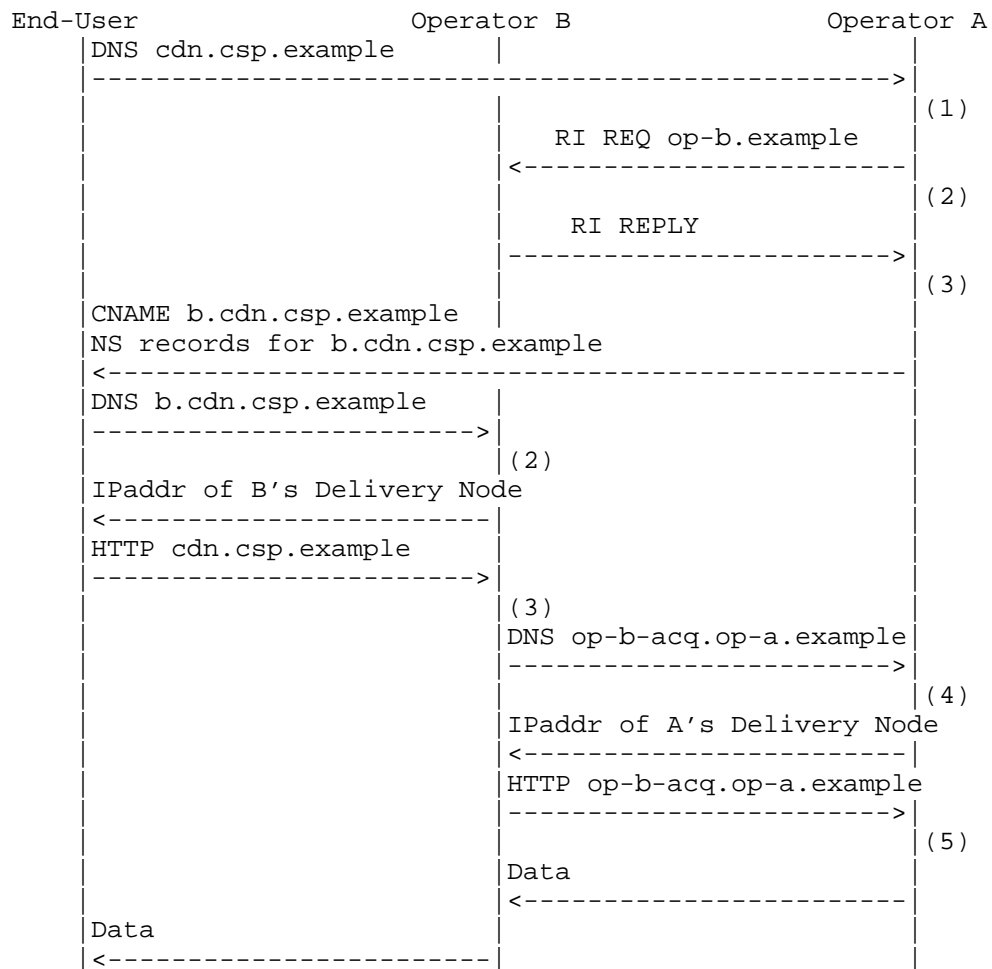


Figure 6: Message Flow for Dynamic Footprint Discovery

This example differs from the one in Figure 5 only in the addition of a RI request (step 2) and corresponding response (step 3). The RI REQ could be a message such as "Can you serve clients from this IP Prefix?" or it could be "Provide the list of client IP prefixes you can currently serve". In either case the response might be cached by operator A to avoid repeatedly asking the same question. Alternatively, or in addition, Operator B may spontaneously advertise to Operator A information (or changes) on the set of requests it is willing and able to serve on behalf of operator A; in that case, Operator B may spontaneously issue RR/RI REPLY messages that are not in direct response to a corresponding RR/RI REQ message. (Note that

the issues of determining the client's subnet from DNS requests, as described above, are exactly the same here as in Section 3.4.)

Once Operator A obtains the RI response, it is now able to determine that Operator B's CDN is an appropriate dCDN for this request and therefore a valid candidate dCDN to consider in its Redirection decision. If that dCDN is selected, the redirection and serving of the request proceeds as before (i.e. in the absence of dynamic footprint discovery).

3.6. Content Removal Example

The following example illustrates how the CDNI Control interface may be used to achieve pre-positioning of an item of content in the dCDN. In this example, user requests for a particular content, and corresponding redirection of such requests from Operator A to Operator B CDN, may (or may not) have taken place earlier. Then, at some point in time, the uCDN (for example, in response to a corresponding Trigger from the Content Provider) uses the CI to request that content identified by a particular URL be removed from dCDN. The following diagram illustrates the operation. It should be noted that a uCDN will typically not know whether a dCDN has cached a given content item, however, it may send the content removal request to make sure no cached versions remain to satisfy any contractual obligations it may have.

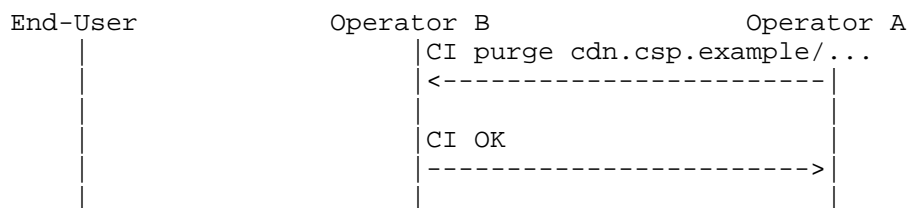


Figure 7: Message Flow for Content Removal

The CI is used to convey the request from uCDN to dCDN that some previously acquired content should be deleted. The URL in the request specifies which content to remove. This example corresponds to a DNS-based redirection scenario such as Section 3.4. If HTTP-based redirection had been used, the URL for removal would be of the form peer-a.op-b.example/cdn.csp.example/...

The dCDN is expected to confirm to the uCDN, as illustrated by the CI OK message, the completion of the removal of the targeted content from all the caches in dCDN.

3.7. Pre-Positioned Content Acquisition Example

The following example illustrates how the CI may be used to pre-position an item of content in the dCDN. In this example, Operator A uses the CDNI Metadata interface to request that content identified by a particular URL be pre-positioned into Operator B CDN.

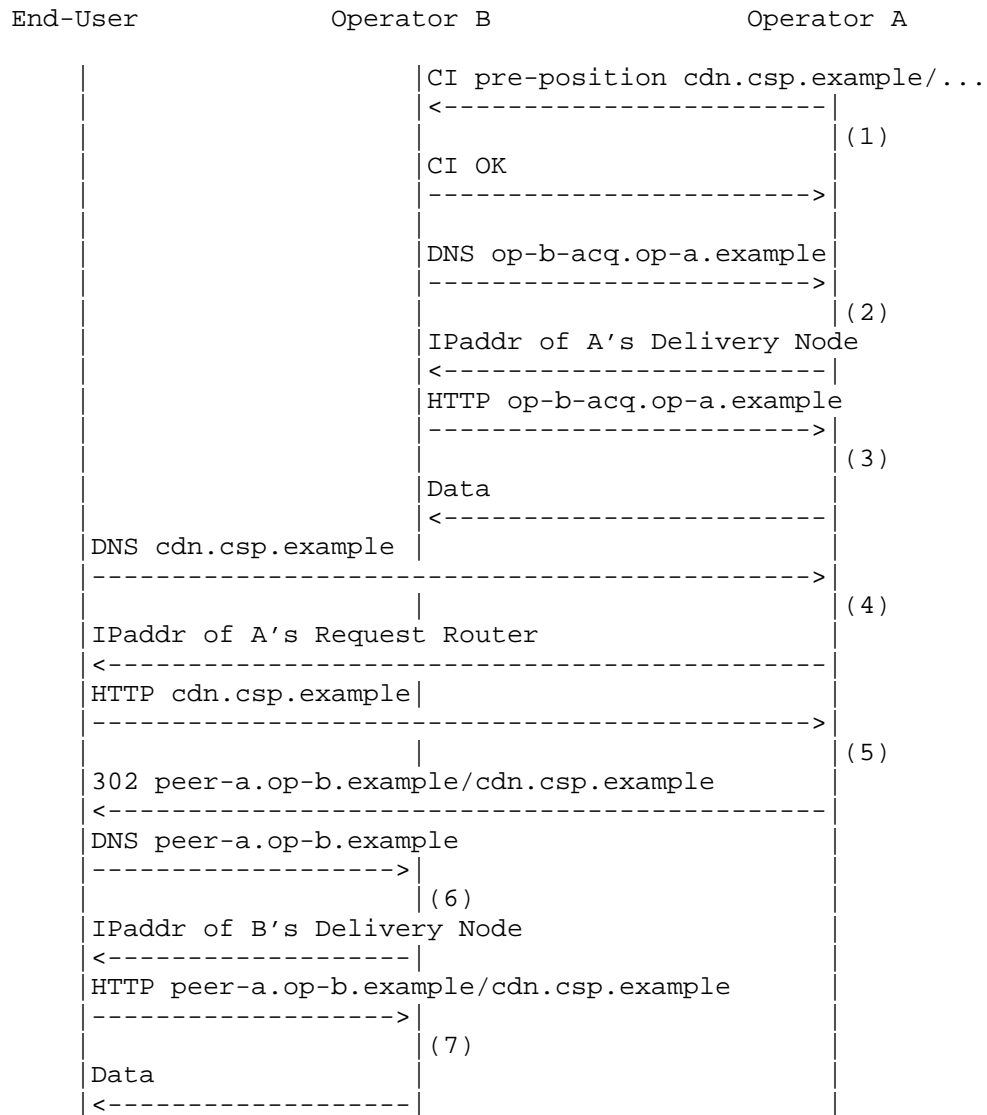


Figure 8: Message Flow for Content Pre-Positioning

The steps illustrated in the figure are as follows:

1. Operator A uses the CI to request that Operator B pre-positions a particular content item identified by its URL. Operator B responds by confirming that it is willing to perform this operation.

Steps 2 and 3 are exactly the same as steps 5 and 6 of Figure 3, only this time those steps happen as the result of the Pre-positioning request instead of as the result of a cache miss.

Steps 4, 5, 6, 7 are exactly the same as steps 1, 2, 3, 4 of Figure 3, only this time Operator B CDN can serve the end-user request without triggering dynamic content acquisition, since the content has been pre-positioned in dCDN. Note that, depending on dCDN operations and policies, the content pre-positioned in the dCDN may be pre-positioned to all, or a subset of, dCDN caches. In the latter case, intra-CDN dynamic content acquisition may take place inside the dCDN serving requests from caches on which the content has not been pre-positioning; however, such intra-CDN dynamic acquisition would not involve the uCDN.

3.8. Asynchronous CDNI Metadata Example

In this section we walk through a simple example illustrating a scenario of asynchronously exchanging CDNI metadata, where the downstream CDN obtains CDNI metadata for content ahead of a corresponding content request. The example that follows assumes that HTTP-based inter-CDN redirection and recursive CDNI request-routing are used, as in Section 3.3. However, Asynchronous exchange of CDNI Metadata is similarly applicable to DNS-based inter-CDN redirection and iterative request routing (in which cases the CDNI metadata may be used at slightly different processing stages of the message flows).

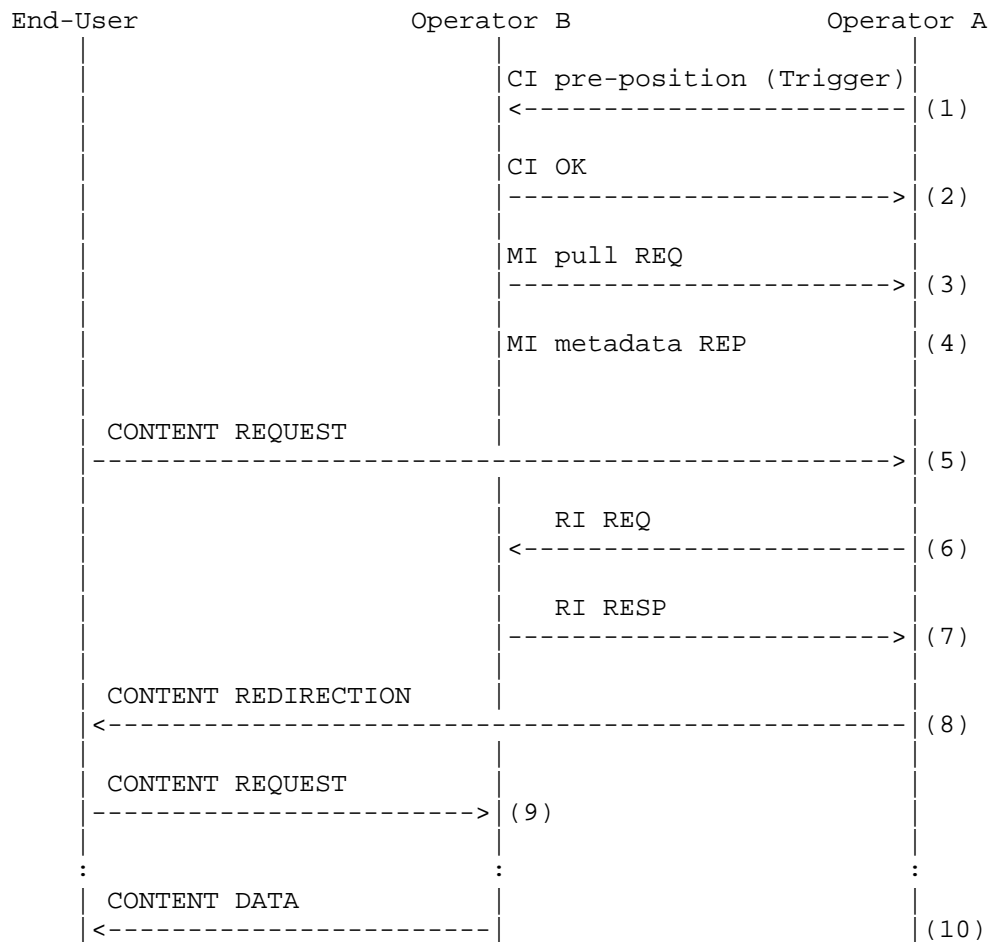


Figure 9: Message Flow for Asynchronous CDNI Metadata

The steps illustrated in the figure are as follows:

1. Operator A uses the CI to Trigger to signal the availability of CDNI metadata to Operator B.
2. Operator B acknowledges the receipt of this Trigger.
3. Operator B requests the latest metadata from Operator A using the MI.
4. Operator A replies with the requested metadata. This document does not constrain how the CDNI metadata information is actually

represented. For the purposes of this example, we assume that Operator A provides CDNI metadata to Operator B indicating that:

- * this CDNI Metadata is applicable to any content referenced by some CDN-Domain.
- * this CDNI metadata consists of a distribution policy requiring enforcement by the delivery node of a specific per-request authorization mechanism (e.g. URI signature or token validation).

5. A Content Request occurs as usual.
6. A CDNI Request Routing Redirection request (RI REQ) is issued by operator A CDN, as discussed in Section 3.3. Operator B's request router can access the CDNI Metadata that are relevant to the requested content and that have been pre-positioned as per Steps 1-4, which may or may not affect the response.
7. Operator B's request router issues a CDNI Request Routing Redirection response (RI RESP) as in Section 3.3.
8. Operator B performs content redirection as discussed in Section 3.3.
9. On receipt of the Content Request by the end user, the delivery node detects that previously acquired CDNI metadata is applicable to the requested content. In accordance with the specific CDNI metadata of this example, the delivery node will invoke the appropriate per-request authorization mechanism, before serving the content. (Details of this authorization are not shown.)
10. Assuming successful per-request authorization, serving of Content Data (possibly preceded by inter-CDN acquisition) proceeds as in Section 3.3.

3.9. Synchronous CDNI Metadata Acquisition Example

In this section we walk through a simple example illustrating a scenario of Synchronous CDNI metadata acquisition, in which the downstream CDN obtains CDNI metadata for content at the time of handling a first request for the corresponding content. As in the preceding section, this example assumes that HTTP-based inter-CDN redirection and recursive CDNI request-routing are used (as in Section 3.3), but dynamic CDNI metadata acquisition is applicable to other variations of request routing.

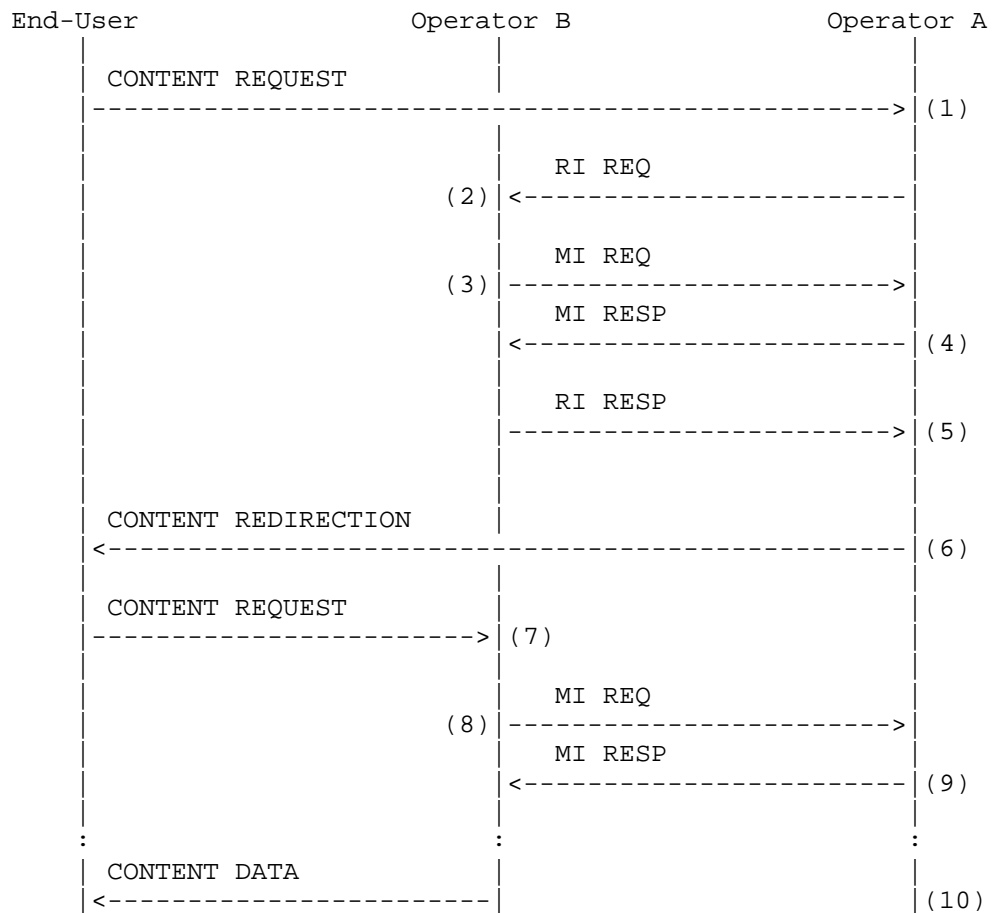


Figure 10: Message Flow for Synchronous CDNI Metadata Acquisition

The steps illustrated in the figure are as follows:

1. A Content Request arrives as normal.
2. An RI request occurs as in the prior example.
3. On receipt of the CDNI Request Routing Request, Operator B's CDN initiates Synchronous acquisition of CDNI Metadata that are needed for routing of the end-user request. We assume the URI for the a Metadata server is known ahead of time through some out-of-band means.

4. On receipt of a CDNI Metadata Request, Operator A's CDN responds, making the corresponding CDNI metadata information available to Operator B's CDN. This metadata is considered by operator B's CDN before responding to the Request Routing request. (In a simple case, the metadata could simply be an allow or deny response for this particular request.)
5. Response to the RI request as normal.
6. Redirection message is sent to the end user.
7. A delivery node of Operator B receives the end user request.
8. The delivery node Triggers dynamic acquisition of additional CDNI metadata that are needed to process the end-user content request. Note that there may exist cases where this step need not happen, for example because the metadata were already acquired previously.
9. Operator A's CDN responds to the CDNI Metadata Request and makes the corresponding CDNI metadata available to Operator B. This metadata influence how Operator B's CDN processes the end-user request.
10. Content is served (possibly preceded by inter-CDN acquisition) as in Section 3.3.

3.10. Content and Metadata Acquisition with Multiple Upstream CDNs

A single dCDN may receive end-user requests from multiple uCDNs. When a dCDN receives an end-user request, it must determine the identity of the uCDN from which it should acquire the requested content.

Ideally, the acquisition path of an end-user request will follow the redirection path of the request. The dCDN should acquire the content from the same uCDN which redirected the request.

Determining the acquisition path requires the dCDN to reconstruct the redirection path based on information in the end-user request. The method for reconstructing the redirection path differs based on the redirection approach: HTTP or DNS.

With HTTP-redirection, the rewritten URI should include sufficient information for the dCDN to directly or indirectly determine the uCDN when the end-user request is received. The HTTP-redirection approach can be further broken-down based on the how the URL is rewritten during redirection: HTTP-redirection with or without Site

Aggregation. HTTP-redirection with Site Aggregation hides the identity of the original CSP. HTTP-redirection without Site Aggregation does not attempt to hide the identity of the original CSP. With both approaches, the rewritten URI includes enough information to identify the immediate neighbor uCDN.

With DNS-redirection, the dCDN receives the published URI (instead of a rewritten URI) and does not have sufficient information for the dCDN to identify the appropriate uCDN. The dCDN may narrow the set of viable uCDNs by examining the CDNI metadata from each to determine which uCDNs are hosting metadata for the requested content. If there is a single uCDN hosting metadata for the requested content, the dCDN can assume that the request redirection is coming from this uCDN and can acquire content from that uCDN. If there are multiple uCDNs hosting metadata for the requested content, the dCDN may be ready to trust any of these uCDNs to acquire the content (provided the uCDN is in a position to serve it). If the dCDN is not ready to trust any of these uCDNs, it needs to ensure via out of band arrangements that, for a given content, only a single uCDN will ever redirect requests to the dCDN.

Content acquisition may be preceded by content metadata acquisition. If possible, the acquisition path for metadata should also follow the redirection path. Additionally, we assume metadata is indexed based on rewritten URIs in the case of HTTP-redirection and is indexed based on published URIs in the case of DNS-redirection. Thus, the RI and the MI are tightly coupled in that the result of request routing (a rewritten URI pointing to the dCDN) serves as an input to metadata lookup. If the content metadata includes information for acquiring the content, then the MI is also tightly coupled with the acquisition interface in that the result of the metadata lookup (an acquisition URL likely hosted by the uCDN) should serve as input to the content acquisition.

4. Main Interfaces

Figure 1 illustrates the main interfaces that are in scope for the CDNI WG, along with several others. The detailed specifications of these interfaces are left to other documents, but see [RFC6707] and [I-D.ietf-cdni-requirements] for some discussion of the interfaces.

One interface that is not shown in Figure 1 is the interface between the user and the CSP. While for the purposes of CDNI that interface is out of scope, it is worth noting that it does exist and can provide useful functions, such as end-to-end performance monitoring and some forms of authentication and authorization.

There is also an important interface between the user and the Request Routing function of both uCDN and dCDN (shown as the "Request" Interface in Figure 1). As we saw in some of the preceding examples, that interface can be used as a way of passing metadata, such as the minimum information that is required for dCDN to obtain the content from uCDN.

In this section we will provide an overview of the functions performed by each of the CDNI interfaces and discuss how they fit into the overall solution. We also examine some of the design tradeoffs, and explore several cross-interface concerns. We begin with an examination of one such tradeoff that affects all the interfaces - the use of in-band or out-of-band communication.

4.1. In-Band versus Out-of-Band Interfaces

Before getting to the individual interfaces, we observe that there is a high-level design choice for each, involving the use of existing in-band communication channels versus defining new out-of-band interfaces.

It is possible that the information needed to carry out various interconnection functions can be communicated between peer CDNs using existing in-band protocols. The use of HTTP 302 redirect is an example of how certain aspects of request routing can be implemented in-band (embedded in URIs). Note that using existing in-band protocols does not imply that the CDNI interfaces are null; it is still necessary to establish the rules (conventions) by which such protocols are used to implement the various interface functions.

There are other opportunities for in-band communication beyond HTTP redirects. For example, many of the HTTP directives used by proxy servers can also be used by peer CDNs to inform each other of caching activity. Of these, one that is particularly relevant is the If-Modified-Since directive, which is used with the GET method to make it conditional: if the requested object has not been modified since the time specified in this field, a copy of the object will not be returned, and instead, a 304 (not modified) response will be returned.

4.2. Cross Interface Concerns

Although the CDNI interfaces are largely independent, there are a set of conventions practiced consistently across all interfaces. Most important among these is how resources are named, for example, how the CDNI Metadata and Control interfaces identify the set of resources to which a given directive applies, or the CDNI Logging

interface identifies the set of resources for which a summary record applies.

While in the limit the CDNI interfaces could explicitly identify every individual resource, in practice, they name resource aggregates (sets of URIs) that are to be treated in a similar way. For example, URI aggregates can be identified by a CDN-Domain (i.e., the FQDN at the beginning of a URI) or by a URI-Filter (i.e., a regular expression that matches a subset of URIs contained in some CDN-Doman). In other words, CDN-Domains and URI-Filters provide a uniform means to aggregate sets (and subsets) of URIs for the purpose of defining the scope for some operation in one of the CDNI interfaces.

4.3. Request Routing Interfaces

The Request Routing interface comprises two parts: the Asynchronous interface used by a dCDN to advertize footprint and capabilities (denoted FCI) to a uCDN, allowing the uCDN to decide whether to redirect particular user requests to that dCDN; and the Synchronous interface used by the uCDN to redirect a user request to the dCDN (denoted RI). (These are somewhat analogous to the operations of routing and forwarding in IP.)

As illustrated in Section 3, the RI part of request routing may be implemented in part by DNS and HTTP. Naming conventions may be established by which CDN peers communicate whether a request should be routed or content served.

We also note that RI plays a key role in enabling recursive redirection, as illustrated in Section 3.3. It enables the user to be redirected to the correct delivery node in dCDN with only a single redirection step (as seen by the user). This may be particularly valuable as the chain of interconnected CDNs increases beyond two CDNs. For further discussion on the RI, see [I-D.ietf-cdni-redirection].

In support of these redirection requests, it is necessary for CDN peers to exchange additional information with each other, and this is the role of the FCI part of request routing. Depending on the method(s) supported, this might include:

- o The operator's unique id (operator-id) or distinguished CDN-Domain (operator-domain);
- o NS records for the operator's set of externally visible request routers;

- o The set of requests the dCDN operator is prepared to serve (e.g. a set of client IP prefixes or geographic regions that may be served by dCDN).
- o Additional capabilities of the dCDN, such as its ability to support different CDNI Metadata requests.

Note that the set of requests that dCDN is willing to serve could in some cases be relatively static (e.g., a set of IP prefixes) which could be exchanged off-line, or might even be negotiated as part of a peering agreement. However, it may also be more dynamic, in which case the exchange supported by FCI would be helpful. A further discussion of the Footprint & Capability Advertisement interface can be found in [I-D.ietf-cdni-footprint-capabilities-semantics].

4.4. CDNI Logging Interface

It is necessary for the upstream CDN to have visibility into the delivery of content that it redirected to a downstream CDN. This allows the upstream CDN to properly bill its customers for multiple deliveries of content cached by the downstream CDN, as well as to report accurate traffic statistics to those content providers. This is one role of the LI.

Other operational data that may be relevant to CDNI can also be exchanged by the LI. For example, dCDN may report the amount of content it has acquired from uCDN, and how much cache storage has been consumed by content cached on behalf of uCDN.

Traffic logs are easily exchanged off-line. For example, the following traffic log is a small deviation from the Apache log file format, where entries include the following fields:

- o Domain - the full domain name of the origin server
- o IP address - the IP address of the client making the request
- o End time - the ending time of the transfer
- o Time zone - any time zone modifier for the end time
- o Method - the transfer command itself (e.g., GET, POST, HEAD)
- o URL - the requested URL
- o Version - the protocol version, such as HTTP/1.0
- o Response - a numeric response code indicating transfer result

- o Bytes Sent - the number of bytes in the body sent to the client
- o Request ID - a unique identifier for this transfer
- o User agent - the user agent, if supplied
- o Duration - the duration of the transfer in milliseconds
- o Cached Bytes - the number of body bytes served from the cache
- o Referer - the referrer string from the client, if supplied

Of these, only the Domain field is indirect in the downstream CDN--it is set to the CDN-Domain used by the upstream CDN rather than the actual origin server. This field could then be used to filter traffic log entries so only those entries matching the upstream CDN are reported to the corresponding operator. Further discussion of the LI can be found in [I-D.ietf-cdni-logging].

One open question is who does the filtering. One option is that the downstream CDN filters its own logs, and passes the relevant records directly to each upstream peer. This requires that the downstream CDN knows the set of CDN-Domains that belong to each upstream peer. If this information is already exchanged between peers as part of another interface, then direct peer-to-peer reporting is straightforward. If it is not available, and operators do not wish to advertise the set of CDN-Domains they serve to their peers, then the second option is for each CDN to send both its non-local traffic records and the set of CDN-Domains it serves to an independent third-party (i.e., a CDN Exchange), which subsequently filters, merges, and distributes traffic records on behalf of each participating CDN operator.

A second open question is how timely traffic information should be. For example, in addition to offline traffic logs, accurate real-time traffic monitoring might also be useful, but such information requires that the downstream CDN inform the upstream CDN each time it serves upstream content from its cache. The downstream CDN can do this, for example, by sending a conditional HTTP GET request (If-Modified-Since) to the upstream CDN each time it receives an HTTP GET request from one of its end-users. This allows the upstream CDN to record that a request has been issued for the purpose of real-time traffic monitoring. The upstream CDN can also use this information to validate the traffic logs received later from the downstream CDN.

There is obviously a tradeoff between accuracy of such monitoring and the overhead of the downstream CDN having to go back to the upstream CDN for every request.

Another design tradeoff in the LI is the degree of aggregation or summarization of data. One situation that lends itself to summarization is the delivery of HTTP adaptive streaming (HAS), since the large number of individual chunk requests potentially results in large volumes of logging information. This case is discussed below, but other forms of aggregation may also be useful. For example, there may be situations where bulk metrics such as bytes delivered per hour may suffice rather than the detailed per-request logs outlined above. It seems likely that a range of granularities of logging will be needed along with ways to specify the type and degree of aggregation required.

4.5. CDNI Control Interface

The CDNI Control interface is initially used to bootstrap the other interfaces. As a simple example, it could be used to provide the address of the logging server in dCDN to uCDN in order to bootstrap the CDNI Logging interface. It may also be used, for example, to establish security associations for the other interfaces.

The other role the CI plays is to allow the uCDN to pre-position, revalidate, or purge metadata and content on a dCDN. These operations, sometimes collectively called the Trigger interface, are discussed further in [I-D.ietf-cdni-control-triggers].

4.6. CDNI Metadata Interface

The role of the CDNI Metadata interface is to enable CDNI distribution metadata to be conveyed to the downstream CDN by the upstream CDN. Such metadata includes geo-blocking restrictions, availability windows, access control policies, and so on. It may also include information to facilitate acquisition of content by dCDN (e.g., alternate sources for the content, authorization information needed to acquire the content from the source). For a full discussion of the CDNI Metadata Interface, see [I-D.ietf-cdni-metadata]

Some distribution metadata may be partially emulated using in-band mechanisms. For example, in case of any geo-blocking restrictions or availability windows, the upstream CDN can elect to redirect a request to the downstream CDN only if that CDN's advertised delivery footprint is acceptable for the requested URL. Similarly, the request could be forwarded only if the current time is within the availability window. However, such approaches typically come with shortcomings such as inability to prevent from replay outside the time window or inability to make use of a downstream CDN that covers a broader footprint than the geo-blocking restrictions.

Similarly, some forms of access control may also be performed on a per-request basis using HTTP directives. For example, being able to respond to a conditional GET request gives the upstream CDN an opportunity to influence how the downstream CDN delivers its content. Minimally, the upstream CDN can invalidate (purge) content previously cached by the downstream CDN.

All of these in-band techniques serve to illustrate that uCDNs have the option of enforcing some of their access control policies themselves (at the expense of increased inter-CDN signaling load), rather than delegating enforcement to dCDNs using the MI. As a consequence, the MI could provide a means for the uCDN to express its desire to retain enforcement for itself. For example, this might be done by including a "check with me" flag in the metadata associated with certain content. The realization of such in-band techniques over the various inter-CDN acquisition protocols (e.g., HTTP) requires further investigation and may require small extensions or semantic changes to the acquisition protocol.

4.7. HTTP Adaptive Streaming Concerns

We consider HTTP Adaptive Streaming (HAS) and the impact it has on the CDNI interfaces because large objects (e.g., videos) are broken into a sequence of small, independent chunks. For each of the following, a more thorough discussion, including an overview of the tradeoffs involved in alternative designs, can be found in RFC 6983.

First, with respect to Content Acquisition and File Management, which are out-of-scope for the CDNI interfaces but nonetheless relevant to the overall operation, we assume no additional measures are required to deal with large numbers of chunks. This means that the dCDN is not explicitly made aware of any relationship between different chunks and the dCDN handles each chunk as if it were an individual and independent content item. The result is that content acquisition between uCDN and dCDN also happens on a per-chunk basis. This approach is in line with the recommendations made in RFC 6983, which also identifies potential improvements in this area that might be considered in the future.

Second, with respect to Request Routing, we note that HAS manifest files have the potential to interfere with request routing since manifest files contain URLs pointing to the location of content chunks. To make sure that a manifest file does not hinder CDNI request routing and does not place excessive load on CDNI resources, the use of manifest files could either be limited to those containing relative URLs or the uCDN could modify the URLs in the manifest. Our approach for dealing with these issues is twofold. As a mandatory requirement, CDNs should be able to handle unmodified manifest files

containing either relative or absolute URLs. To limit the number of redirects, and thus the load placed on the CDNI interfaces, as an optional feature uCDNs can use the information obtained through the CDNI Request Routing Redirection interface to modify the URLs in the manifest file. Since the modification of the manifest file is an optional uCDN-internal process, this does not require any standardization effort beyond being able to communicate chunk locations in the CDNI Request Routing Redirection interface.

Third, with respect to the CDNI Logging interface, there are several potential issues, including the large number of individual chunk requests potentially resulting in large volumes of logging information, and the desire to correlate logging information for chunk requests that correspond to the same HAS session. For the initial CDNI specification, our approach is to expect participating CDNs to support per-chunk logging (e.g. logging each chunk request as if it were an independent content request) over the CDNI Logging interface. Optionally, the LI may include a Content Collection Identifier (CCID) and/or a Session Identifier (SID) as part of the logging fields, thereby facilitating correlation of per-chunk logs into per-session logs for applications benefiting from such session level information (e.g. session-based analytics). This approach is in line with the recommendations made in RFC 6983, which also identifies potential improvements in this area that might be considered in the future.

Fourth, with respect to the CDNI Control interface, and in particular purging HAS chunks from a given CDN, our approach is to expect each CDN supports per-chunk content purge (e.g. purging of chunks as if they were individual content items). Optionally, a CDN may support content purge on the basis of a "Purge Identifier (Purge-ID)" allowing the removal of all chunks related to a given Content Collection with a single reference. It is possible that this Purge-ID could be merged with the CCID discussed above for HAS Logging, or alternatively, they may remain distinct.

4.8. URI Rewriting

When using HTTP redirection, content URIs may be rewritten when redirection takes place within an uCDN, from an uCDN to a dCDN, and within the dCDN. In the case of cascaded CDNs, content URIs may be rewritten at every CDN hop (e.g., between the uCDN and the dCDN acting as the transit CDN, and between the transit CDN and the dCDN serving the request. The content URI used between any uCDN/dCDN pair becomes a common handle that can be referred to without ambiguity by both CDNs in all their inter-CDN communications. This handle allows the uCDN and dCDN to correlate information exchanged using other CDNI

interfaces in both the downstream direction (e.g., when using the MI) and the upstream direction (e.g., when using the LI).

Consider the simple case of a single uCDN/dCDN pair using HTTP redirection. We introduce the following terminology for content URIs to simplify the discussion:

"u-URI" represents a content URI in a request presented to the uCDN;

"ud-URI" is a content URI acting as the common handle across uCDN and dCDN for requests redirected by the uCDN to a specific dCDN;

"d-URI" represents a content URI in a request made within the delegate dCDN.

In our simple pair-wise example, the "ud-URI" effectively becomes the handle that the uCDN/dCDN pair use to correlate all CDNI information. In particular, for a given pair of CDNs executing the HTTP redirection, the uCDN needs to map the u-URI to the ud-URI handle for all MI message exchanges, while the dCDN needs to map the d-URI to the ud-URI handle for all LI message exchanges.

In the case of cascaded CDNs, the transit CDN will rewrite the content URI when redirecting to the dCDN, thereby establishing a new handle between the transit CDN and the dCDN, that is different from the handle between the uCDN and transit CDN. It is the responsibility of the transit CDN to manage its mapping across handles so the right handle for all pairs of CDNs is always used in its CDNI communication.

In summary, all CDNI interfaces between a given pair of CDNs need to always use the "ud-URI" handle for that specific CDN pair as their content URI reference.

5. Deployment Models

In this section we describe a number of possible deployment models that may be achieved using the CDNI interfaces described above. We note that these models are by no means exhaustive, and that many other models may be possible.

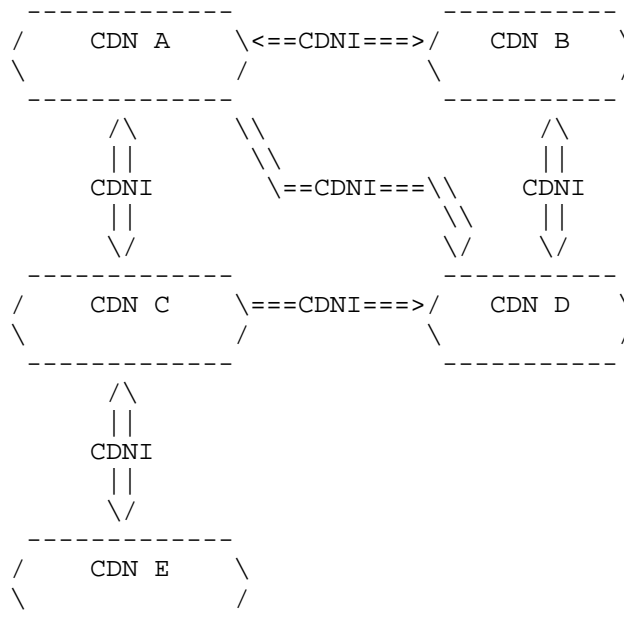
Although the reference model of Figure 1 shows all CDN functions on each side of the CDNI interface, deployments can rely on entities that are involved in any subset of these functions, and therefore only support the relevant subset of CDNI interfaces. As already noted in Section 3, effective CDNI deployments can be built without

necessarily implementing all the interfaces. Some examples of such deployments are shown below.

Note that, while we refer to upstream and downstream CDNs, this distinction applies to specific content items and transactions. That is, a given CDN may be upstream for some transactions and downstream for others, depending on many factors such as location of the requesting client and the particular piece of content requested.

5.1. Meshed CDNs

Although the reference model illustrated in Figure 1 shows a unidirectional CDN interconnection with a single uCDN and a single dCDN, any arbitrary CDNI meshing can be built from this, such as the example meshing illustrated in Figure 11. (Support for arbitrary meshing may or may not be in the initial scope for the working group, but the model allows for it.)



- ====> CDNI interfaces, with right-hand side CDN acting as dCDN to left-hand side CDN
- <====> CDNI interfaces, with right-hand side CDN acting as dCDN to left-hand side CDN and with left-hand side CDN acting as dCDN to right-hand side CDN

Figure 11: CDNI Deployment Model: CDN Meshing Example

5.2. CSP combined with CDN

Note that our terminology refers to functional roles and not economic or business roles. That is, a given organization may be operating as both a CSP and a fully fledged uCDN when we consider the functions performed, as illustrated in Figure 12.

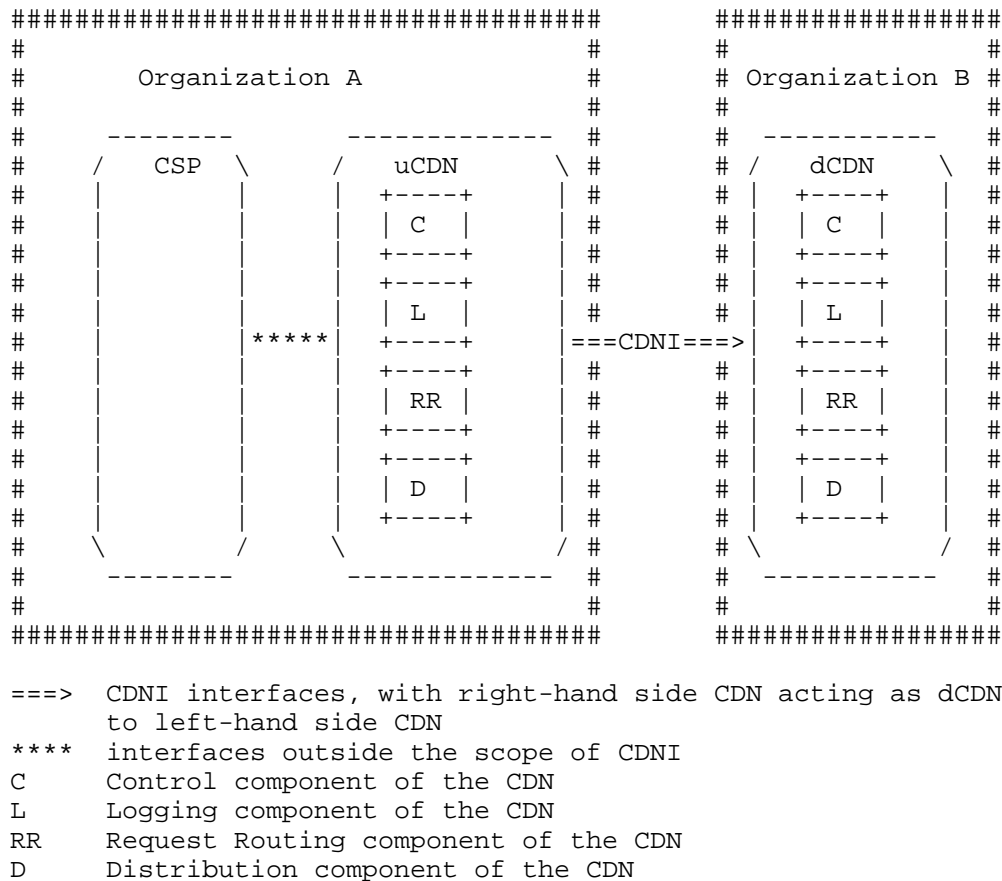


Figure 12: CDNI Deployment Model: Organization combining CSP & uCDN

5.3. CSP using CDNI Request Routing Interface

As another example, a content provider organization may choose to run its own request routing function as a way to select among multiple candidate CDN providers; In this case the content provider may be modeled as the combination of a CSP and of a special, restricted case of a CDN. In that case, as illustrated in Figure 13, the CDNI Request Routing interfaces can be used between the restricted CDN

operated by the content provider Organization and the CDN operated by the full CDN organization acting as a dCDN in the request routing control plane. Interfaces outside the scope of the CDNI work can be used between the CSP functional entities of the content provider organization and the CDN operated by the full CDN organization acting as a uCDN) in the CDNI control planes other than the request routing plane (i.e. Control, Distribution, Logging).

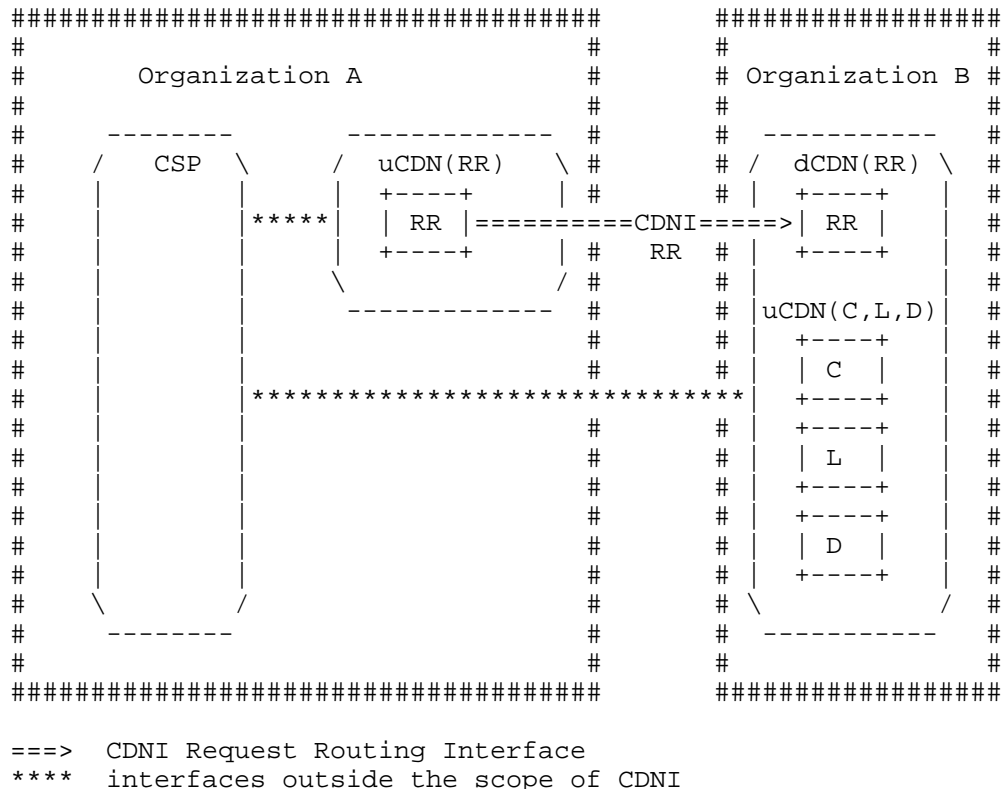


Figure 13: CDNI Deployment Model: Organization combining CSP and partial CDN

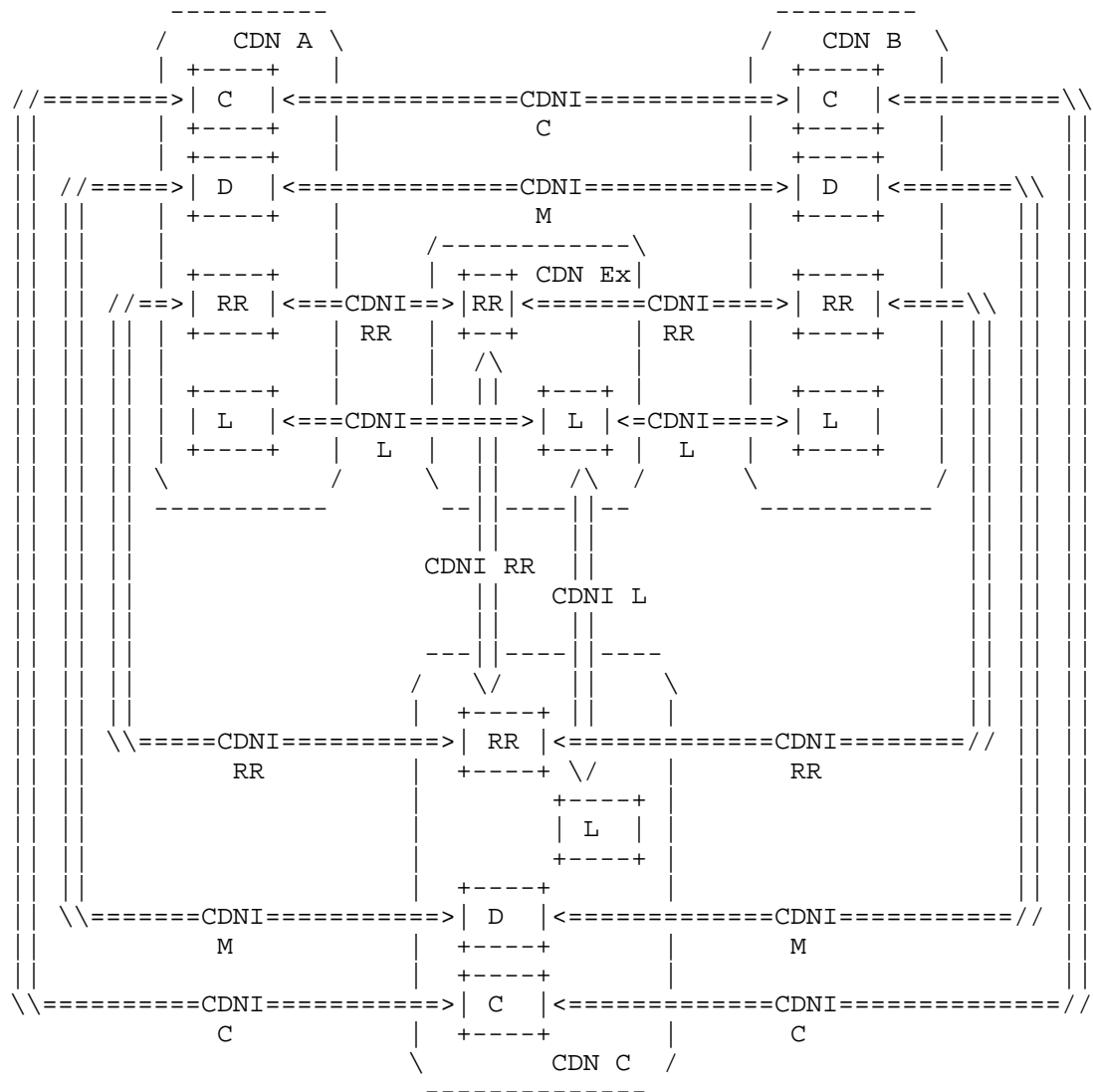
5.4. CDN Federations and CDN Exchanges

There are two additional concepts related to, but distinct from CDN Interconnection. The first is CDN Federation. Our view is that CDNI is the more general concept, involving two or more CDNs serving content to each other's users, while federation implies a multi-lateral interconnection arrangement, but other CDN interconnection agreements are also possible (e.g., symmetric bilateral, asymmetric bilateral). An important conclusion is that CDNI technology should

not presume (or bake in) a particular interconnection agreement, but should instead be general enough to permit alternative interconnection arrangements to evolve.

The second concept often used in the context of CDN Federation is CDN Exchange--a third party broker or exchange that is used to facilitate a CDN federation. Our view is that a CDN exchange offers valuable machinery to scale the number of CDN operators involved in a multi-lateral (federated) agreement, but that this machinery is built on top of the core CDNI interconnection mechanisms. For example, as illustrated in Figure 14, the exchange might aggregate and redistribute information about each CDN footprint and capacity, as well as collect, filter, and redistribute traffic logs that each participant needs for interconnection settlement, but inter-CDN request routing, inter-CDN content distribution (including inter-CDN acquisition) and inter-CDN control which fundamentally involve a direct interaction between an upstream CDN and a downstream CDN--operate exactly as in a pair-wise peering arrangement. Turning to Figure 14, we observe that in this example:

- o each CDN supports a direct CDNI Control interface to every other CDN
- o each CDN supports a direct CDNI Metadata interface to every other CDN
- o each CDN supports a CDNI Logging interface with the CDN Exchange
- o each CDN supports both a CDNI Request Routing interface with the CDN Exchange (for aggregation and redistribution of dynamic CDN footprint discovery information) and a direct RI to every other CDN (for actual request redirection).



<=CDNI RR=> CDNI Request Routing Interface
 <=CDNI M==> CDNI Metadata Interface
 <=CDNI C==> CDNI Control Interface
 <=CDNI L==> CDNI Logging Interface

Figure 14: CDNI Deployment Model: CDN Exchange

Note that a CDN exchange may alternatively support a different set of functionality (e.g. Logging only, or Logging and full request

routing, or all the functionality of a CDN including content distribution). All these options are expected to be allowed by the IETF CDNI specifications.

6. Trust Model

There are a number of trust issues that need to be addressed by a CDNI solution. Many of them are in fact similar or identical to those in a simple CDN without interconnection. In a standard CDN environment (without CDNI), the CSP places a degree of trust in a single CDN operator to perform many functions. The CDN is trusted to deliver content with appropriate quality of experience for the end user. The CSP trusts the CDN operator not to corrupt or modify the content. The CSP often relies on the CDN operator to provide reliable accounting information regarding the volume of delivered content. The CSP may also trust the CDN operator to perform actions such as timely invalidation of content and restriction of access to content based on certain criteria such as location of the user and time of day, and to enforce per-request authorization performed by the CSP using techniques such as URI signing.

A CSP also places trust in the CDN not to distribute any information that is confidential to the CSP (e.g., how popular a given piece of content is) or confidential to the end user (e.g., which content has been watched by which user).

A CSP does not necessarily have to place complete trust in a CDN. A CSP will in some cases take steps to protect its content from improper distribution by a CDN, e.g. by encrypting it and distributing keys in some out of band way. A CSP also depends on monitoring (possibly by third parties) and reporting to verify that the CDN has performed adequately. A CSP may use techniques such as client-based metering to verify that accounting information provided by the CDN is reliable. HTTP conditional requests may be used to provide the CSP with some checks on CDN operation. In other words, while a CSP may trust a CDN to perform some functions in the short term, the CSP is able in most cases to verify whether these actions have been performed correctly and to take action (such as moving the content to a different CDN) if the CDN does not live up to expectations.

One of the trust issues raised by CDNI is transitive trust. A CDN that has a direct relationship with a CSP can now "outsource" the delivery of content to another (downstream) CDN. That CDN may in turn outsource delivery to yet another downstream CDN, and so on.

The top level CDN in such a chain of delegation is responsible for ensuring that the requirements of the CSP are met. Failure to do so

is presumably just as serious as in the traditional single CDN case. Hence, an upstream CDN is essentially trusting a downstream CDN to perform functions on its behalf in just the same way as a CSP trusts a single CDN. Monitoring and reporting can similarly be used to verify that the downstream CDN has performed appropriately. However, the introduction of multiple CDNs in the path between CSP and end user complicates the picture. For example, third party monitoring of CDN performance (or other aspects of operation, such as timely invalidation) might be able to identify the fact that a problem occurred somewhere in the chain but not point to the particular CDN at fault.

In summary, we assume that an upstream CDN will invest a certain amount of trust in a downstream CDN, but that it will verify that the downstream CDN is performing correctly, and take corrective action (including potentially breaking off its relationship with that CDN) if behavior is not correct. We do not expect that the trust relationship between a CSP and its "top level" CDN will differ significantly from that found today in single CDN situations. However, it does appear that more sophisticated tools and techniques for monitoring CDN performance and behavior will be required to enable the identification of the CDN at fault in a particular delivery chain.

We expect that the detailed designs for the specific interfaces for CDNI will need to take the transitive trust issues into account. For example, explicit confirmation that some action (such as content removal) has taken place in a downstream CDN may help to mitigate some issues of transitive trust.

7. IANA Considerations

This memo includes no request to IANA.

8. Privacy Considerations

In general, a CDN has the opportunity to collect detailed information about the behavior of end-users e.g. by logging which files are being downloaded. While the concept of interconnected CDNs as described in this document doesn't necessarily allow any given CDN to gather more information on any specific user, it potentially facilitates sharing of this data by a CDN with more parties. As an example, the purpose of the CDNI Logging Interface is to allow a dCDN to share some of its log records with a uCDN, both for billing purposes as well as for sharing traffic statistics with the Content Provider on which behalf the content was delivered. The fact that the CDNI Interfaces provide mechanisms for sharing such potentially sensitive user data, shows that it is necessary to include in these interface appropriate

privacy and confidentiality mechanisms. The definition of such mechanisms is dealt with in the respective CDN interface documents.

9. Security Considerations

While there are a variety of security issues introduced by a single CDN, we are concerned here specifically with the additional issues that arise when CDNs are interconnected. For example, when a single CDN has the ability to distribute content on behalf of a CSP, there may be concerns that such content could be distributed to parties who are not authorized to receive it, and there are mechanisms to deal with such concerns. Our focus in this section is on how CDN interconnection introduces new security issues not found in the single CDN case. For a more detailed analysis of the security requirements of CDNI, see section 9 of [I-D.ietf-cdni-requirements].

Many of the security issues that arise in CDNI are related to the transitivity of trust (or lack thereof) described in Section 6. As noted above, the design of the various interfaces for CDNI must take account of the additional risks posed by the fact that a CDN with whom a CSP has no direct relationship is now potentially distributing content for that CSP. The mechanisms used to mitigate these risks may be similar to those used in the single CDN case, but their suitability in this more complex environment must be validated.

CDNs today offer a variety of means to control access to content, such as time-of-day restrictions, geo-blocking, and URI signing. These mechanisms must continue to function in CDNI environments, and this consideration is likely to affect the design of certain CDNI interfaces (e.g. metadata, request routing). For more information on URI signing in CDNI, see [I-D.leung-cdni-uri-signing].

Just as with a single CDN, each peer CDN must ensure that it is not used as an "open proxy" to deliver content on behalf of a malicious CSP. Whereas a single CDN typically addresses this problem by having CSPs explicitly register content (or origin servers) that are to be served, simply propagating this information to peer downstream CDNs may be problematic because it reveals more information than the upstream CDN is willing to specify. (To this end, the content acquisition step in the earlier examples force the dCDN to retrieve content from the uCDN rather than go directly to the origin server.)

There are several approaches to this problem. One is for the uCDN to encode a signed token generated from a shared secret in each URL routed to a dCDN, and for the dCDN to validate the request based on this token. Another one is to have each upstream CDN advertise the set of CDN-Domains they serve, where the downstream CDN checks each request against this set before caching and delivering the associated

object. Although straightforward, this approach requires operators to reveal additional information, which may or may not be an issue.

9.1. Security of CDNI Interfaces

It is noted in [I-D.ietf-cdni-requirements] that all CDNI interfaces must be able to operate securely over insecure IP networks. Since it is expected that the CDNI interfaces will be implemented using existing application protocols such as HTTP or XMPP, we also expect that the security mechanisms available to those protocols may be used by the CDNI interfaces. Details of how these interfaces are secured will be specified in the relevant interface documents.

9.2. Digital Rights Management

Issues of digital rights management (DRM, also sometimes called digital restrictions management) is often employed for content distributed via CDNs. In general, DRM relies on the CDN to distribute encrypted content, with decryption keys distributed to users by some other means (e.g. directly from the CSP to the end user.) For this reason, DRM is considered out of scope [RFC6707] and does not introduce additional security issues for CDNI.

10. Contributors

The following individuals contributed to this document:

- o Matt Caulfield
- o Francois le Faucheur
- o Aaron Falk
- o David Ferguson
- o John Hartman
- o Ben Niven-Jenkins
- o Kent Leung

11. Acknowledgements

The authors would like to thank Huw Jones and Jinmei Tatuya for their helpful input to this document. In addition, the authors would like to thank Stephen Farrell, Ted Lemon and Alissa Cooper for their reviews, which have helped to improve this document.

12. Informative References

- [I-D.ietf-cdni-control-triggers]
Murray, R. and B. Niven-Jenkins, "CDNI Control Interface / Triggers", draft-ietf-cdni-control-triggers-02 (work in progress), December 2013.
- [I-D.ietf-cdni-footprint-capabilities-semantic]
Seedorf, J., Peterson, J., Previdi, S., Brandenburg, R., and K. Ma, "CDNI Request Routing: Footprint and Capabilities Semantics", draft-ietf-cdni-footprint-capabilities-semantic-02 (work in progress), February 2014.
- [I-D.ietf-cdni-logging]
Faucheur, F., Bertrand, G., Oprescu, I., and R. Peterkofsky, "CDNI Logging Interface", draft-ietf-cdni-logging-11 (work in progress), March 2014.
- [I-D.ietf-cdni-metadata]
Niven-Jenkins, B., Murray, R., Watson, G., Caulfield, M., Leung, K., and K. Ma, "CDN Interconnect Metadata", draft-ietf-cdni-metadata-06 (work in progress), February 2014.
- [I-D.ietf-cdni-redirection]
Niven-Jenkins, B. and R. Brandenburg, "Request Routing Redirection Interface for CDN Interconnection", draft-ietf-cdni-redirection-02 (work in progress), April 2014.
- [I-D.ietf-cdni-requirements]
Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", draft-ietf-cdni-requirements-17 (work in progress), January 2014.
- [I-D.leung-cdni-uri-signing]
Leung, K., Faucheur, F., Downey, B., Brandenburg, R., and S. Leibrand, "URI Signing for CDN Interconnection (CDNI)", draft-leung-cdni-uri-signing-05 (work in progress), March 2014.
- [RFC3466] Day, M., Cain, B., Tomlinson, G., and P. Rzewski, "A Model for Content Internetworking (CDI)", RFC 3466, February 2003.
- [RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", RFC 6707, September 2012.

- [RFC6770] Bertrand, G., Stephan, E., Burbridge, T., Eardley, P., Ma, K., and G. Watson, "Use Cases for Content Delivery Network Interconnection", RFC 6770, November 2012.
- [RFC6983] van Brandenburg, R., van Deventer, O., Le Faucheur, F., and K. Leung, "Models for HTTP-Adaptive-Streaming-Aware Content Distribution Network Interconnection (CDNI)", RFC 6983, July 2013.

Authors' Addresses

Larry Peterson
Akamai Technologies, Inc.
8 Cambridge Center
Cambridge, MA 02142
USA

Email: lapeters@akamai.com

Bruce Davie
VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
USA

Email: bdavie@vmware.com

Ray van Brandenburg (editor)
TNO
Brassersplein 2
Delft 2612CT
the Netherlands

Phone: +31-88-866-7000
Email: ray.vanbrandenburg@tno.nl

CDNI
Internet-Draft
Intended status: Informational
Expires: December 21, 2012

W. Jin
M. Li
B. Khasnabish
ZTE Corporation
June 19, 2012

Content De-duplication for CDNi Optimization
draft-jin-cdni-content-deduplication-optimization-01

Abstract

Recent explosive growth of content delivery/distribution networks (CDNs) and their interconnection are causing unintended repetition of content storage in the same dCDN. This can be avoided by using a suitable de-duplication mechanism. This document explores the scenarios which create the problems, and then discusses the approaches to eliminate the duplicated transmission of the same content from uCDN(s) to dCDN in CDNi networks. To implement the optimization, some enhancements to the CDNi metadata model and interface are required.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 21, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Terminology	3
2. Deployment Scenarios	4
2.1. Scenario 1	4
2.2. Scenario 2	5
2.3. Scenario 3	6
3. Content Naming for CDNi	7
3.1. Uniqueness	7
3.2. Ownership	8
4. CDNi Content De-duplication Optimization Implementation	8
4.1. Constant URL	8
4.2. Content Naming Mechanism	9
4.3. Description of Content De-duplication	10
4.3.1. Pre-Positioned Content Acquisition	10
4.3.2. Dynamic Content Acquisition	12
4.3.3. Content Removal	14
5. Security Considerations	15
6. IANA Considerations	15
7. Acknowledgments	16
8. References	16
8.1. Normative References	16
8.2. Informative References	16
Authors' Addresses	16

1. Introduction

In some CDNi deployment, the dCDN occasionally caches the same content copy multiple times from the same Content Service Provider (CSP). For example, the CSP may have the agreement with two authoritative CDNs, and both could be the upstream CDNs of the same dCDN. Cascading of CDNs may result in a similar scenario as well. The top-layer uCDN establishes connections with two intermediate-layer uCDNs respectively, and both connect to the same bottom-layer dCDN. In such scenarios, the dCDN may receive the content via 'push' from one of the uCDN via pre-position procedure, and then may also request for downloading the same content from another uCDN via another pre-position procedure or upon user's content request.

Storing the same content multiple times not only wastes the dCDN's the memory or storage resources, it also causes wastage of transmission bandwidth that is used to deliver the same content repeatedly. Therefore, it is necessary to avoid delivering the same content from different uCDNs to dCDN repeatedly. In this draft, we list a set of scenarios which may cause repeated delivery of the same content. A feasible solution for content de-duplication is then discussed.

In order to address the content repetition problem, several issues need to be considered.

- * How to detect content repetition by dCDN.
- * How to avoid content repetition, when one or more uCDNs selects one dCDN to deliver the same content to multiple User Agents.

This document provides detailed analysis on the issues of content repetition. We realize that there is a need to develop an optimized mechanism to de-duplicate the content in CDNi network. In order to implement such optimization, enhancement to CDNi metadata model and interface may be required.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

This document reuses the terminology defined in:

[draft-ietf-cdni-problem-statement-06],

[draft-ietf-cdni-requirements-03],

[draft-ietf-cdni-framework-00], and

[draft-ietf-cdni-use-cases-07].

Resource Id: a metadata object (e.g., partial or whole URL, or other format) which is generated by uCDN and identifies the storage of the content in the uCDN.

Content Id: a metadata object (e.g. a URN) which uniquely identifies the content in the scope of CDNi.

2. Deployment Scenarios

This section illustrates several CDNi deployment scenarios that typically lead to duplicated content in the same server.

2.1. Scenario 1

As depicted in Figure 1, two interconnected CDNs - CDN-A(uCDN) and CDN-B(dCDN) - both have contracts with CSP. CDN-B plays two roles at the same time: downstream CDN of CDN-A and Authoritative CDN of CSP. When an end-user of CDN-A initiates content request from the CSP, CDN-A decides that CDN-B should be the serving CDN. Then CDN-A redirects the request to CDN-B. If CDN B does not have a local copy of the requested content yet (cache miss), CDN B ingests the content from CDN A. Normally CDN-B, as Authoritative CDN, is very likely to have already cached this content from original server. If CDN-B cannot identify the requested contents as the same content, this same content will be repeatedly retrieved and cached.

As the location of the content in a CDN is normally assigned by CDN itself, the URLs of the same content are likely different between CDNs. So it is not enough to determine whether the content to be retrieved and cached is the same only by the URL of the content.

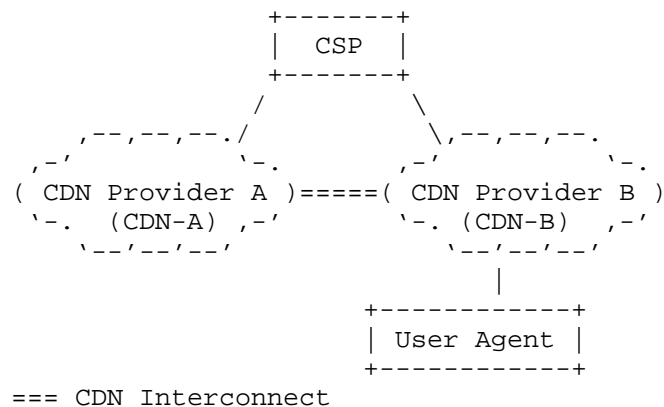


Figure 1 Interconnected CDNs with the same CSP

2.2. Scenario 2

As depicted in Figure 2, both CDN-A and CDN-B establish interconnections with CDN-C that acts as a dCDN. Thus, CDN-C will cache the content for CDN-A and CDN-B. When both CDN provider A and CDN provider B have agreements with the same CSP for content delivery, CDN-C may be required by CDN-A and CDN-B separately to retrieve and cache the same content from the CSP. Similar to Scenario 1, CDN-C is also likely to suffer from content repetition troubles.

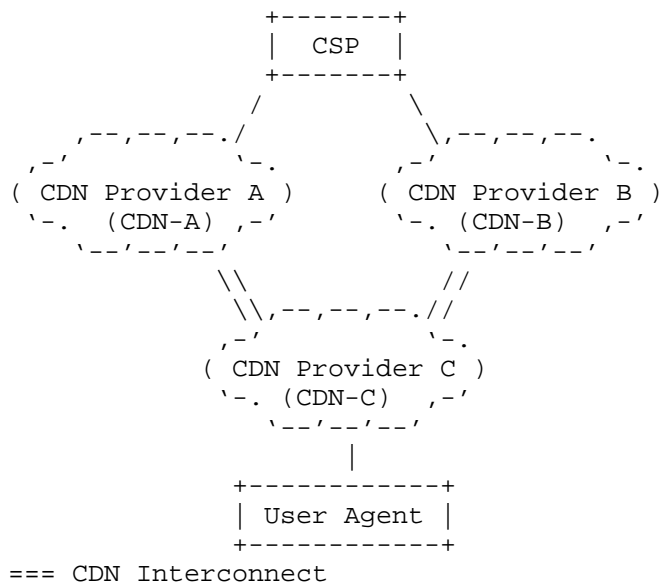


Figure 2 Interconnected CDNs with the same CSP and with one dCDN

2.3. Scenario 3

Now we consider the case of cascaded CDNs, as depicted in Figure 3. Note that the top-layer Upstream CDN-A has direct contract with CSP and interconnects with two middle-layer CDNs (CDN-B and CDN-C) that have the same bottom-layer Downstream CDN (CDN-D) interconnected to them. Consequently, there are two possible delivery paths for CDN-D to cache the contents of CSP, one is CDN-A -> CDN-B -> CDN-D, and the other is CDN-A -> CDN-C -> CDN-D. CDN-D may need to cache the same content by upstream CDNs (CDN-B and CDN-C) on different paths. If the URL of the content is changed by CDN-B or CDN-C, CDN-D cannot be aware of the contents to be cached and therefore this may lead to storing of duplicated contents.

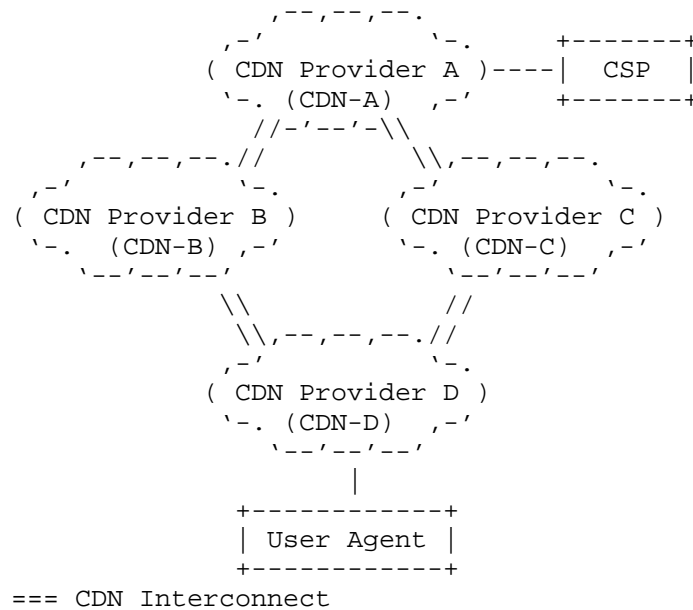


Figure 3 Cascaded CDNs

3. Content Naming for CDNi

It is well known that CDNs have their own content naming mechanisms, most of which are independent and separated from one another due to the use of different algorithms such as Hash algorithms. It implies that for the same content distributed by two CDNs, the corresponding content identifiers are likely to be quite different.

[draft-ietf-cdni-requirements-03] treats the information regarding CDN content naming as intra-CDN information and the CDNI solution MUST not require intra-CDN information to be exposed to other CDNs for effective and efficient delivery of the content. Therefore, establishing a uniform content naming mechanism is urgently needed for CDNi network. This mechanism which can be implemented by CDNI Metadata Distribution Protocol may have the following properties.

3.1. Uniqueness

CDNi content naming mechanism must guarantee the uniqueness of the content identification. Although URL is widely used for identifying network resource, it is not quite suitable for content identification in CDNi network where content de-duplication needs to be taken into consideration. Although the method of URL match is commonly used by many cache systems to detect the repetitive files with same name for

avoiding content repetition, it will probably fail for CDNi case. This is due to the fact that different forwarding mechanisms are used in different CDNs involved. The user-originated requests are always snooped by the DPI (deep packet inspection) devices before transmitted to the original server, whereas the requests received by dCDN are always redirected by one or more uCDNs. Since there is no guarantee that the URLs will not be changed through the redirection process, a new type of object needs to be defined to represent the uniqueness of content identifier.

For the CSP's the contents that are distributed into different interconnected CDNs, the related metadata objects may be somewhat different in many cases. For example, in Figure 2, when both CDN-A and CDN-B delegate the delivery of the same CSP's content, the content metadata such as (a) content description, (b) access policy, and (c) security policy may be not be exactly the same in both cases. Such metadata information is not suitable for content identification. Consequently, we need to define a new type of metadata object that helps uniquely identify the same content.

3.2. Ownership

CDNi content naming mechanism should embody the ownership of content identification. Typically, a CDN provider may have contracts with many CSPs for delivering their contents, as well as operate its own Content delivery network. However, it may happen that a lot of contents published by these CSPs may be very similar, and many of them may even be exactly the same. Therefore, the problem is whether these identical copies are from the same CSP or how the interconnected CDNs can verify that these contents are identical. (Note: Such copies are pointed by the same content identification only if they are from the same content source.) For a traditional (non-interconnected) CDN, there is no problem to distinguish them via its intra content naming mechanism. When a CDN interconnects with other CDNs, the condition becomes more complicated due to the lack of awareness of CSP's content when the CDN acts as a dCDN.

4. CDNi Content De-duplication Optimization Implementation

4.1. Constant URL

In general, URL-based mechanism can be used to implement content de-duplication in CDNi network. As referred in section 2, the URL description for a content may be different from uCDN to uCDN and is likely to be changed in the redirection process. An agreement to configure a specific URL between a pair of interconnected CDN is used in the draft [draft-ietf-cdni-framework-00], however this method is

not flexible enough for supporting de-duplication in complex CDNi network. So a feasible proposal is to specify a mechanism for CDNi network to guarantee that CSP's same contents cached in different uCDNs are identified by the same URL and that the URL is unchanged or at least the path part remains the same in the redirection process. The main problem of this mechanism is lack of resilience and we prefer an alternative mechanism as introduced in section 4.2 below.

4.2. Content Naming Mechanism

This section provides a detailed description of CDNi content naming mechanism using CDNi Metadata Protocol.

In general, CSP's content as well as its copies cached in interconnected CDNs are delivered to numerous End-Users. The draft [draft-ietf-cdni-framework-00] assigns each copy an identifier in the form of an URL that is embedded with "CDN-Domain" which is used to distinguish whether a download request is from an end-user or from an uCDN. We use the term Resource Identifier to represent the storage pointing to the content copies in interconnected CDNs. However, unlike the usage in the draft [draft-ietf-cdni-framework-00], in this document CDNi content naming mechanism specifies Resource Identifiers as the one which is only related to contents in uCDNs. Taking the example in section 2.3, we use Resource Identifier A to point to content originated through uCDN A.

Although the Resource Identifier is able to identify a content, the uniqueness of content identification can't be guaranteed, as Resource Identifier is used to identify the storage of the content in the uCDN and therefore will be changed during redirection processes between different uCDNs. In order to resolve it, we introduce the term Content Identifier that is assigned to associate with Resource Identifier to uniquely identify the content and is similar to the URN usage. Note that the Content Identifier MUST be globally unique. Figure 4 shows a metadata model that can be used for maintaining the relationship between the two types of Identifiers. Using this model, the dCDN is able to uniquely identify and route requests towards the same targeted content.

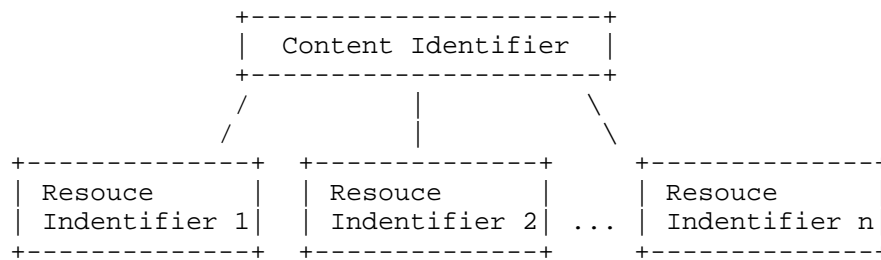


Figure 4 Metadata Model for Maintaining Relationship among Multi-IDs

Note: We need to develop an authoritative 'Entity' for creating and maintaining the Content Identifier.

4.3. Description of Content De-duplication

In this section the details of the solutions that use CDNi Content Naming mechanism for content de-duplication are discussed.

By using the content identification model included in content metadata, an interconnected CDN is able to detect content repetition. The content status must be synchronously updated by the interconnected CDN. According to content status, the interconnected CDN can determine whether the resource copy is cached or not.

We present several procedures for optimized implementation of avoidance of CDNi content repetition.

4.3.1. Pre-Positioned Content Acquisition

The following flow illustrates how the two uCDNs successively pre-position the same content in the dCDN. In this flow, the content to be pre-positioned in the dCDN is identified by different Resource Identifiers corresponding to uCDN A and uCDN B.

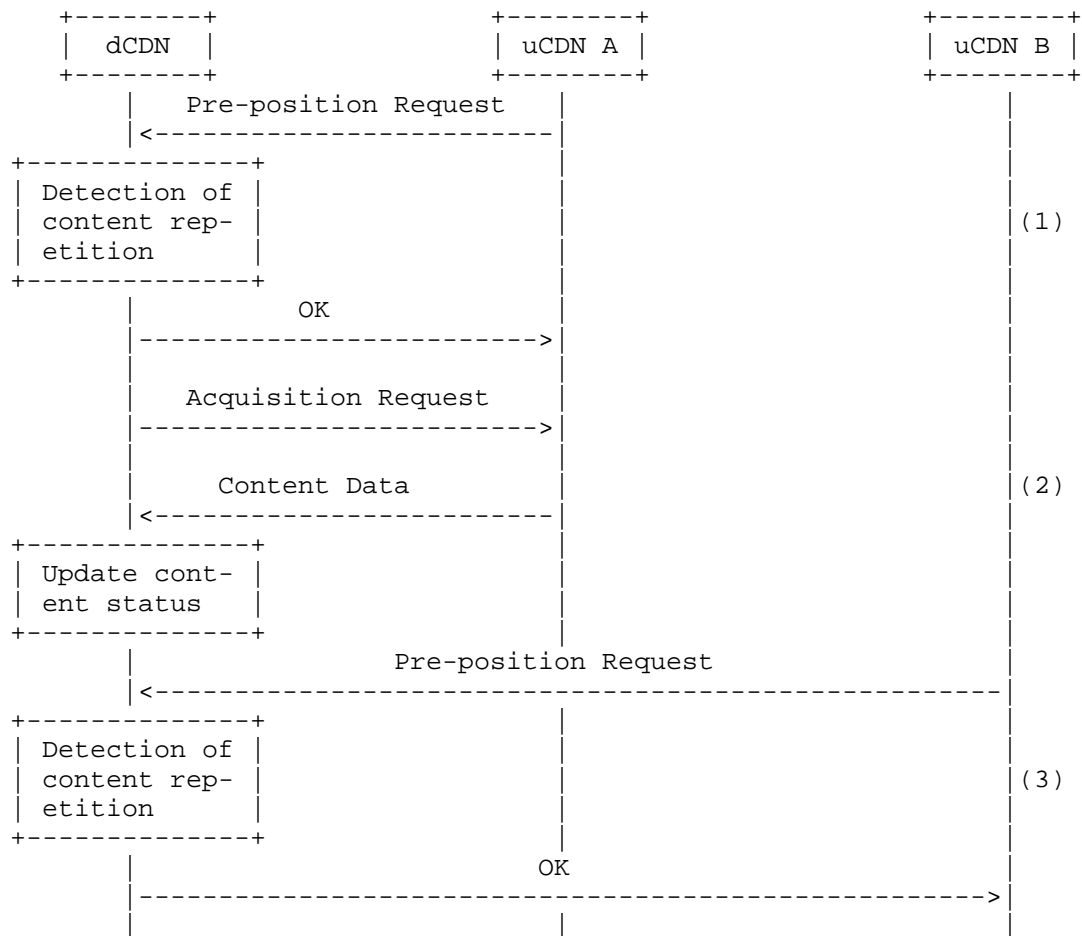


Figure 5 Acquisition of Pre-Positioned Content

The steps that are illustrated in the figure are as follows:

1. The uCDN A requests that the dCDN pre-positions a particular content item identified by its Resource Identifier and Content Identifier. Receiving the message, the dCDN uses the Content Identifier to look up target metadata to see whether the same content item is cached. With the result that the metadata does not exist, the dCDN replies to uCDN A with an OK message to notify that no such copy has been cached and content pre-position is required.
2. The dCDN acquires the content from uCDN A. Once the content is pre-positioned, dCDN updates the content status and maintains the

binding relation between Resource Identifier and Content Identifier metadata.

3. The uCDN B requests that dCDN pre-positions the same content item identified by its Resource Identifier and Content Identifier. Receiving the message, the dCDN uses the Content Identifier to look up target metadata. Because such metadata exists, dCDN determines that the content is already cached. Then, dCDN replies to uCDN A with an OK message to notify that the same copy has been cached and the pre-position request should be cancelled. The dCDN locally binds the new Resource Identifier provided by uCDN B with the Content Identifier.

4.3.2. Dynamic Content Acquisition

The following flows illustrates how the dCDN performs content de-duplication in cases of a cache miss and a cache hit without content pre-positioning.

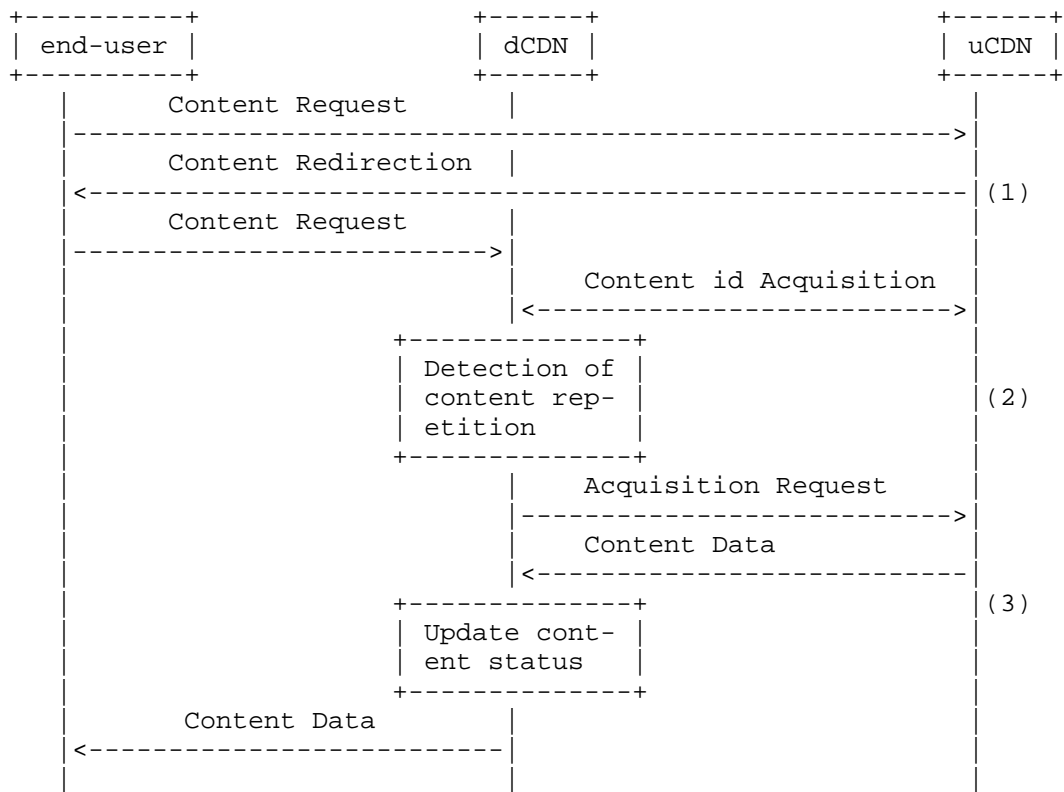


Figure 6 Dynamic Content Acquisition (cache miss case)

The steps that are illustrated in the figure are as follows:

1. A content request originated from an end-user is received at uCDN. The uCDN processes the request and recognizes that the end-user is best served by a dCDN. So uCDN redirects the request to the dCDN by sending redirection response to the end-user who then requests the content from the dCDN. The uCDN encapsulates Resource Identifier of the requested content item in the redirection response.

2. Receiving the request, the dCDN uses the Resource Identifier pointing to the requested resource to fetch the corresponding Content Identifier from the uCDN. The dCDN then uses the Content Identifier to look up target metadata to check whether the content item is cached. With the result that such metadata does not exist, the case of a cache miss is determined by the dCDN, and therefore content needs to be downloaded from the uCDN before delivered to the end-user.

3. The dCDN acquires the requested content from uCDN A. Once the content is cached, dCDN updates the content status and maintains the binding relation between Resource Identifier and the Content Identifier metadata. The dCDN then delivers content data to the end-user.

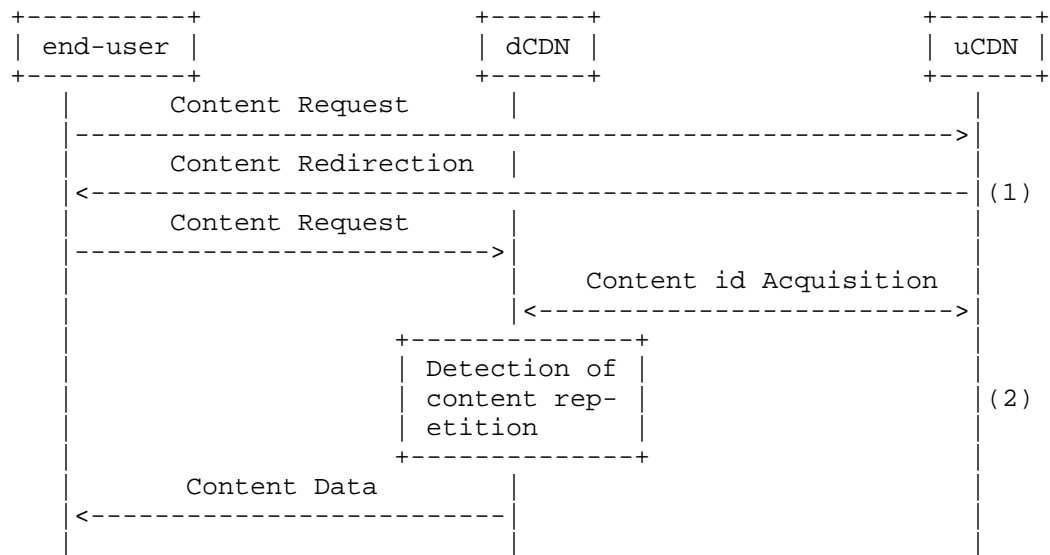


Figure 7 Dynamic Content Acquisition (cache hit case)

The steps that are illustrated in the figure are as follows:

Steps 1 and 2 are exactly the same as steps 1 and 2 of Figure 6, except that this time dCDN determines the case of a cache hit according to the existence of such record in the corresponding Content Identifier metadata.

This flow differs from the one in Figure 6 only in terms of not triggering dynamic content acquisition (step 3), since the content has already been cached by dCDN.

4.3.3. Content Removal

The following flow illustrates how the dCDN removes the content resource under the control of the uCDN.

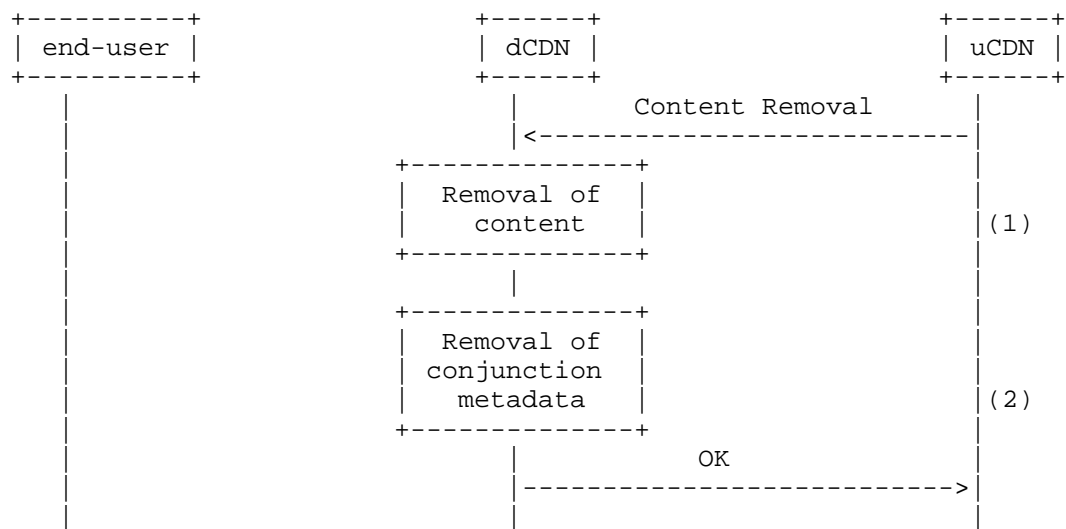


Figure 8 Removal of Content

A premise is that the content copy to be removed has already been cached in the dCDN from the uCDN. The steps illustrated in the figure are as follows:

1. The uCDN requests that the dCDN removes some content resource identified by the Content Identifier due to the deployment policy or expiration of content!_s life-time. The dCDN then removes the content resource accordingly.
2. Once the content resource is removed, the dCDN updates the content status and removes the Content Identifier metadata. It then replies a OK response to the uCDN.

5. Security Considerations

To be added later

6. IANA Considerations

This document has no IANA Considerations.

7. Acknowledgments

To be added later

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. Informative References

- [I-D.ietf-cdni-framework]
Peterson, L. and B. Davie, "Framework for CDN Interconnection", April 2012.
- [I-D.ietf-cdni-problem-statement]
Niven-Jenkins, B., Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", May 2012.
- [I-D.ietf-cdni-requirements]
Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", December 2011.
- [I-D.narten-iana-considerations-rfc2434bis]
Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", draft-narten-iana-considerations-rfc2434bis-09 (work in progress), March 2008.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

Authors' Addresses

WeiYi Jin
ZTE Corporation
Nanjing, 210012
China

Phone: +86 025-52871364
Email: jin.weiya@zte.com.cn

Mian Li
ZTE Corporation
Nanjing, 210012
China

Phone: +86 025-88014641
Email: li.mian@zte.com.cn

Bhumip Khasnabish
ZTE Corporation
New Jersey, 07960
USA

Phone: +001-781-752-8003
Email: bhumip.khasnabish@zteusa.com

CDNI
Internet-Draft
Intended status: Informational
Expires: September 29, 2013

W. Jin
M. Li
B. Khasnabish
ZTE Corporation
March 28, 2013

Content De-duplication for CDNI Optimization
draft-jin-cdni-content-deduplication-optimization-04

Abstract

Recent explosive growth of content delivery/distribution networks (CDNs) and their interconnection are causing unintended repetition of content storage in the same dCDN. This can be avoided by using a suitable de-duplication mechanism. This document explores the scenarios which create the problem, and then discusses the approaches to eliminate the duplicated transmission of the same content from uCDN(s) to dCDN in CDNI networks. To implement the optimization, some enhancement to the CDNI metadata model and interface is required.

We realize that for business-specific purposes the same content may be encrypted/packaged with different keys for different providers. The impact of DRM (Digital Rights Management) technology on de-duplication will be discussed in a future version of this draft.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 29, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Terminology	3
2. Deployment Scenarios	4
2.1. Impact of Content Duplication on the Network System	4
2.2. Current Data Deduplication Technologies	5
2.3. Content Duplication Scenario Involved in this Draft	5
2.3.1. Scenario 1	5
2.3.2. Scenario 2	6
2.3.3. Scenario 3	7
3. Content Naming for CDNi	8
3.1. Uniqueness	8
3.2. Ownership	9
4. CDNi Content De-duplication Optimization Implementation	9
4.1. Constant URL	9
4.2. Content Naming Mechanism	10
4.3. Content ID	11
4.4. Description of Content De-duplication	12
4.4.1. Pre-Positioned Content Acquisition	13
4.4.2. Dynamic Content Acquisition	14
4.4.3. Content Purge and Invalidate	16
5. Security Considerations	18
6. IANA Considerations	18
7. Acknowledgments	18
8. References	18
8.1. Normative References	18
8.2. Informative References	18
Authors' Addresses	19

1. Introduction

In some CDNi deployment, the dCDN occasionally caches the same content copy multiple times from the same Content Service Provider (CSP). For example, the CSP may have the agreement with two authoritative CDNs, and both could be the upstream CDNs of the same dCDN. Cascading of CDNs may result in a similar scenario as well. The top-layer uCDN establishes connections with two intermediate-layer uCDNs respectively, and both connect to the same bottom-layer dCDN. In such scenarios, the dCDN may receive the content via 'push' from one of the uCDN via pre-position procedure, and then may also request for downloading the same content from another uCDN via another pre-position procedure or upon user's content request.

Copies of the same content may be transferred to one dCDN, which results in waste of the dCDN's memory or storage, and transmission bandwidth that is used to deliver the copies repeatedly. Therefore, it is necessary to avoid delivering the same content from different uCDNs to dCDN repeatedly. In this draft, we list a set of scenarios which may cause repeated delivery of the same content. A feasible solution for content de-duplication is then discussed.

In order to address the content repetition problem, several issues need to be considered.

- * How to detect content repetition by dCDN.

- * How to avoid content repetition, when one or more uCDNs select(s) one dCDN to deliver the same content to multiple User Agents.

This document provides detailed analysis on the issues of content repetition. We realize that there is a need to develop an optimized mechanism to de-duplicate the content in CDNi network. In order to implement such optimization, enhancement to CDNi metadata model and interface may be required.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

This document reuses the terminology defined in:

[RFC 6707],

[RFC 6770],

[draft-ietf-cdni-framework], and

[draft-ietf-cdni-requirements].

Resource Id: a metadata object (e.g., partial or whole URL, or other format) which is generated by uCDN and identifies the storage of the content in the uCDN.

Content Id: a metadata object (e.g. a URN) which uniquely identifies the content in the scope of CDNi.

2. Deployment Scenarios

This section illustrates several CDNi deployment scenarios that typically lead to duplicated content in the same server.

2.1. Impact of Content Duplication on the Network System

Along with the explosive growth of digital information, the storage space required by data is increasing as well. In the past decade, the capacity of storage system provided by many industries has developed from dozens of gigabytes to hundreds of terabytes or even more. With the exponential growth of data, enterprises are facing with much more frequent data backup and recovery. The cost to manage and store data, as well as the space and consumption of data center, is also growing into a more and more serious situation. Survey exposes that up to 60% of the data stored in the application system is redundant and the proportion continues to grow along with the time moving forward. This also leads to extra load of the network in transmitting repeat copies of the same content.

As for CDN, the duplication of the content delivered will:

1. increase the complexity of CDN management. An increasing number of content tables need to be maintained, which will reduce the efficiency of content search;
2. demand larger CDN storage capacity, which would be a waste of facility and investment;
3. lead to inaccurate statistics of hot content, which consequently results in the inaccuracy of the arithmetic for the hot content dispatching in the CDN;
4. increase the latency for the CDN content access. Such cases as local content could not be hit and the same content has to be acquired from other CDNs would occur more frequently.

2.2. Current Data Deduplication Technologies

At present, data deduplication technologies are widely used in the storage backup and archiving system, in which the deduplication module is responsible for comparing and analyzing the data content, finding out redundant data and sending corresponding metadata back to the storage service interface. Finally, non-duplicated data will be stored into the storage medium. Main technologies can be divided into:

1. Identical Data Detection: identical data includes identical files and identical data block. Whole File Detection (WFD) technology uses Hash for data mining. Fixed-sized partition (FSP) detection technology, content-defined chunking (CDC) detection technology and sliding block technology are used for duplicated data lookup and deletion;

2. Resembling Data Detection: according to the resemblance characteristics of the data itself, shingle technology, bloom filter technology and mode match technology are used to find out the duplicated data that can not be detected by Identical Data Detection technologies.

Above technologies are applied under the prerequisite that the content is completed downloaded and stored. However, for CDN, comparison of the content after downloading is a waste of transmission traffic and computation. In addition, with a widespread deployment of CDNi system, content duplication issue will not result from above reasons. Instead, it results from the redirection procedures used in CDNi during which URLs pointed to the same content is changed. In this case, dCDN would download the same content several times due to different URLs that in fact point to the same content. Therefore, in CDNi, current data deduplication technologies can not be used to address the issue and scenarios proposed in this draft.

2.3. Content Duplication Scenario Involved in this Draft

In this document, the content duplication results from the redirection procedures used in CDNi during which URLs pointed to the same content is changed. In this case, dCDN would download the same content several times due to different URLs that in fact point to the same content.

2.3.1. Scenario 1

As depicted in Figure 1, both CDN-A and CDN-B establish interconnections with CDN-C that acts as a dCDN. Thus, CDN-C will

cache the content for CDN-A and CDN-B. When both CDN provider A and CDN provider B have agreements with the same CSP for content delivery, CDN-C may be required by CDN-A and CDN-B separately to retrieve and cache the same content from the CSP. CDN-C is likely to suffer from content repetition problems.

As the location of the content in a CDN is normally assigned by CDN itself, the URLs of the same content are likely to be different between CDNs. So it is not enough to determine whether the content to be retrieved and cached is the same only by the URLs of the content.

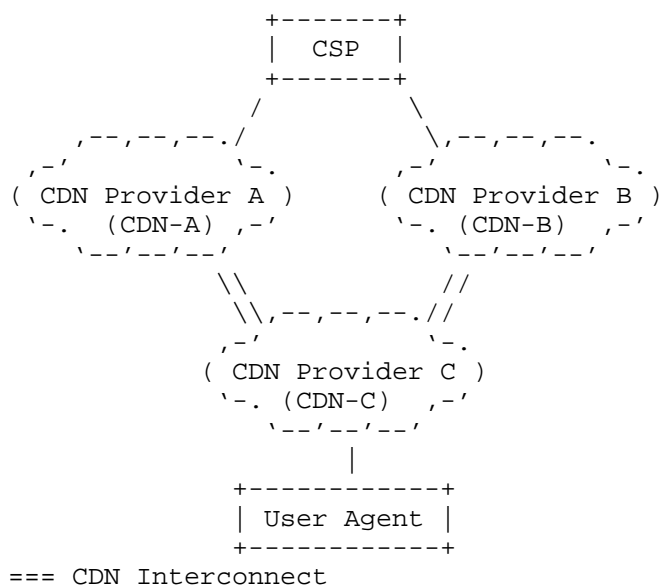


Figure 1 Interconnected CDNs with the same CSP and with one dCDN

2.3.2. Scenario 2

Now we consider the case of cascaded CDNs, as depicted in Figure 2. Note that the top-layer Upstream CDN-A has direct contract with CSP and interconnects with two middle-layer CDNs (CDN-B and CDN-C) that have the same bottom-layer Downstream CDN (CDN-D) interconnected to them. Consequently, there are two possible delivery paths for CDN-D to cache the content of CSP. One is CDN-A -> CDN-B -> CDN-D, and the other is CDN-A -> CDN-C -> CDN-D. CDN-D may cache the same content by upstream CDNs (CDN-B and CDN-C) on different paths. If the URL of the content is changed by CDN-B or CDN-C, CDN-D is not able to be aware of the content to be cached and therefore this may lead to

duplicate storage of the same content.

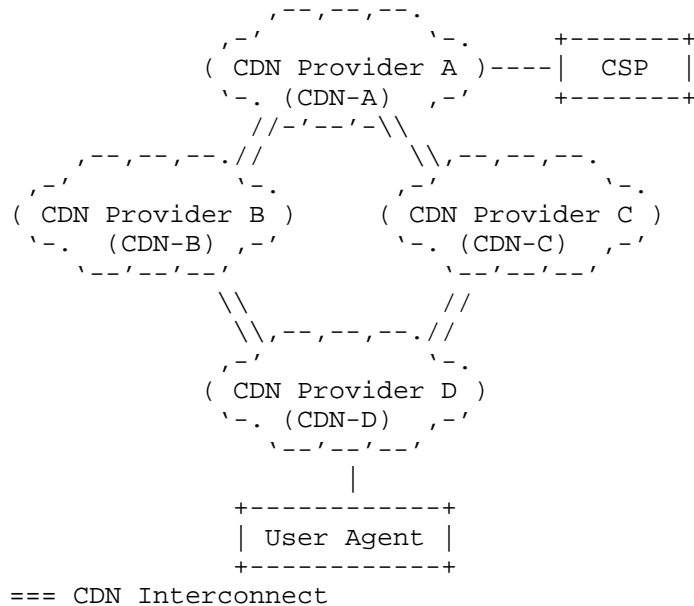


Figure 2 Cascaded CDNs

2.3.3. Scenario 3

As depicted in Figure 3, both two interconnected CDNs - CDN-A(uCDN) and CDN-B(dCDN) - have contracts with CSP. CDN-B plays two roles at the same time: downstream CDN of CDN-A and Authoritative CDN of CSP. When an end-user of CDN-A served by CDN-B initiates content request from the CSP, CDN-A decides that CDN-B should be the serving CDN. Then CDN-A redirects the request to CDN-B. If CDN B does not have a local copy of the requested content (cache miss), CDN B ingests the content from CDN A. When CSP pushes the same content to CDN-B and if CDN-B cannot identify the duplication, this same content will be repeatedly retrieved and cached.

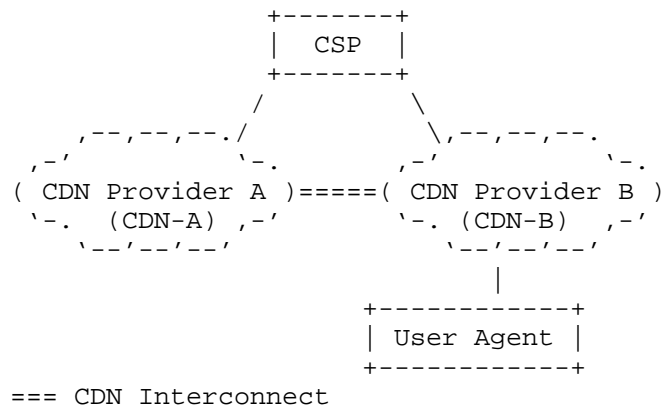


Figure 3 Interconnected CDNs with the same CSP

3. Content Naming for CDNi

It is well known that CDNs have their own content naming mechanisms, most of which are independent and separated from one another due to the use of different algorithms such as Hash algorithms. It implies that for the same content distributed by two CDNs, the corresponding content identifiers are likely to be quite different. [I-D.ietf-cdni-requirements] treats the information regarding CDN content naming as intra-CDN information and the CDNI solution MUST not require intra-CDN information to be exposed to other CDNs for effective and efficient delivery of the content. Therefore, establishing a uniform content naming mechanism is urgently needed for CDNi network. This mechanism which can be implemented by CDNI Metadata Distribution Protocol may have the following properties.

3.1. Uniqueness

CDNi content naming mechanism must guarantee the uniqueness of the content identification. Although URL is widely used for identifying network resource, it is not quite suitable for content identification in CDNi network where content de-duplication needs to be taken into consideration. Although the method of URL match is commonly used by many cache systems to detect the repetitive files with same name for avoiding content repetition, it will probably fail for CDNi case. This is due to the fact that different forwarding mechanisms are used in different CDNs involved. The user-originated requests are always snooped by the DPI (deep packet inspection) devices before transmitted to the original server, whereas the requests received by dCDN are always redirected by one or more uCDNs. Since there is no guarantee that the URLs will not be changed through the redirection

process, a new type of object needs to be defined to represent the uniqueness of content identifier.

For the CSP's the contents that are distributed into different interconnected CDNs, the related metadata objects may be somewhat different in many cases. For example, in Figure 2, when both CDN-A and CDN-B delegate the delivery of the same CSP's content, the content metadata such as (a) content description, (b) access policy, and (c) security policy may be not be exactly the same in both cases. Such metadata information is not suitable for content identification. Consequently, we need to define a new type of metadata object that helps uniquely identify the same content.

3.2. Ownership

CDNi content naming mechanism should embody the ownership of content identification. Typically, a CDN provider may have contracts with many CSPs for delivering their contents, as well as operate its own Content delivery network. However, it may happen that a lot of contents published by these CSPs may be very similar, and many of them may even be exactly the same. Therefore, the problem is whether these identical copies are from the same CSP or how the interconnected CDNs can verify that these contents are identical. (Note: Such copies are pointed by the same content identification only if they are from the same content source.) For a traditional (non-interconnected) CDN, there is no problem to distinguish them via its intra content naming mechanism. When a CDN interconnects with other CDNs, the condition becomes more complicated due to the lack of awareness of CSP's content when the CDN acts as a dCDN.

4. CDNi Content De-duplication Optimization Implementation

4.1. Constant URL

In general, URL-based mechanism can be used to implement content de-duplication in CDNi network. As referred in section 2, the URL description for a content may be different from uCDN to uCDN and is likely to be changed in the redirection process. An agreement to configure a specific URL between a pair of interconnected CDN is used in the draft [I-D.ietf-cdni-framework-01], however this method is not flexible enough for supporting de-duplication in complex CDNi network. So a feasible proposal is to specify a mechanism for CDNi network to guarantee that CSP's same contents cached in different uCDNs are identified by the same URL and that the URL is unchanged or at least the path part remains the same in the redirection process. The main problem of this mechanism is lack of resilience and we prefer an alternative mechanism as introduced in section 4.2 below.

4.2. Content Naming Mechanism

This section provides a detailed description of CDNi content naming mechanism using CDNi Metadata Protocol.

In general, CSP's content as well as its copies cached in interconnected CDNs are delivered to numerous End-Users. The draft [I-D.ietf-cdni-framework-01] assigns each copy an identifier in the form of an URL that is embedded with "CDN-Domain" which is used to distinguish whether a download request is from an end-user or from an uCDN. We use the term Resource Identifier to represent the storage pointing to the content copies in interconnected CDNs. However, unlike the usage in the draft [I-D.ietf-cdni-framework-01], in this document CDNi content naming mechanism specifies Resource Identifiers as the one which is only related to contents in uCDNs. Taking the example in section 2.3, we use Resource Identifier A to point to content originated through uCDN A.

Although the Resource Identifier is able to identify a content, the uniqueness of content identification can't be guaranteed, as Resource Identifier is used to identify the storage of the content in the uCDN and therefore will be changed during redirection processes between different uCDNs. In order to resolve it, we introduce the term Content Identifier that is assigned to associate with Resource Identifier to uniquely identify the content and is similar to the URN usage. Note that the Content Identifier MUST be globally unique. Figure 4 shows a metadata model that can be used for maintaining the relationship between the two types of Identifiers. Using this model, the dCDN is able to uniquely identify and route requests towards the same targeted content.

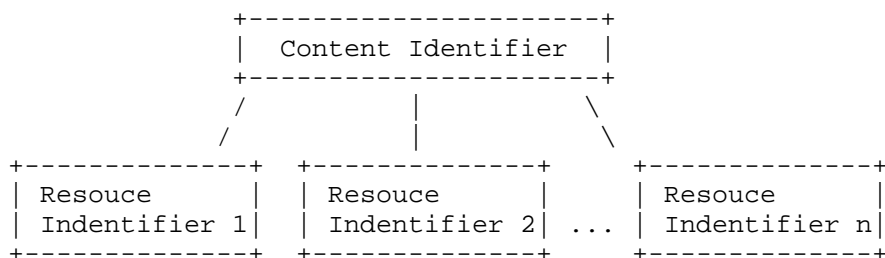


Figure 4 Metadata Model for Maintaining Relationship among Multi-IDs

Note: We need to develop an authoritative 'Entity' for creating and maintaining the Content Identifier.

4.3. Content ID

Actually, EIDR (Entertainment Identifier Registry), an industry non-profit organization, has already started the research and standardization of the globally unique content identifier and its generating mechanism. As stated in the whitepaper of [UNIVERSAL UNIQUE IDENTIFIERS IN MOVIE AND TELEVISION SUPPLY CHAIN MANAGEMENT], EIDR offers an inexpensive mechanism for uniquely identifying the complete range of audiovisual assets relevant to commerce including micro-assets such as clips and newer types of objects such as encodings. The EIDR data model can be readily extended to cover new and emerging objects and relationships as the industry evolves over time. EIDR naming system meets the requirements of coverage, flexibility, extensibility, scalability, cost-effectiveness, interoperability, etc.

The EIDR registry assigns a unique universal identifier for all registered assets. EIDR is an opaque ID with all information about the registered asset stored in the central registry. Its structure consists of a standard registry prefix, the unique suffix for each asset and a check digit. The suffix of an asset ID is of the form XXXX-XXXX-XXXX-XXXX-XXXX-C, where X is a hexadecimal digit and C is the ISO 7064 Mod 37, 36 check character.

Standard Prefix for EIDR Registry	Unique Suffix for each asset	check digit
10.5240/	XXXX-XXXX-XXXX-XXXX-XXXX	-C

Figure 5 Structure of Content ID

The content provider submits content items for registration to the registry system, along with core metadata and information. The system uses a sophisticated system to insure that the object submitted to the registry has not already been registered and then generates an EIDR for the content. All the above need to be done before the content is injected into CDNi system. After that, the content identifier, used as metadata of the content, is injected into CDNi system and transmitted between uCDN and dCDN via such interface as Control Interface, Metadata Interface.

The detail information of EIDR can be referred to in the whitepaper of [UNIVERSAL UNIQUE IDENTIFIERS IN MOVIE AND TELEVISION SUPPLY CHAIN MANAGEMENT].

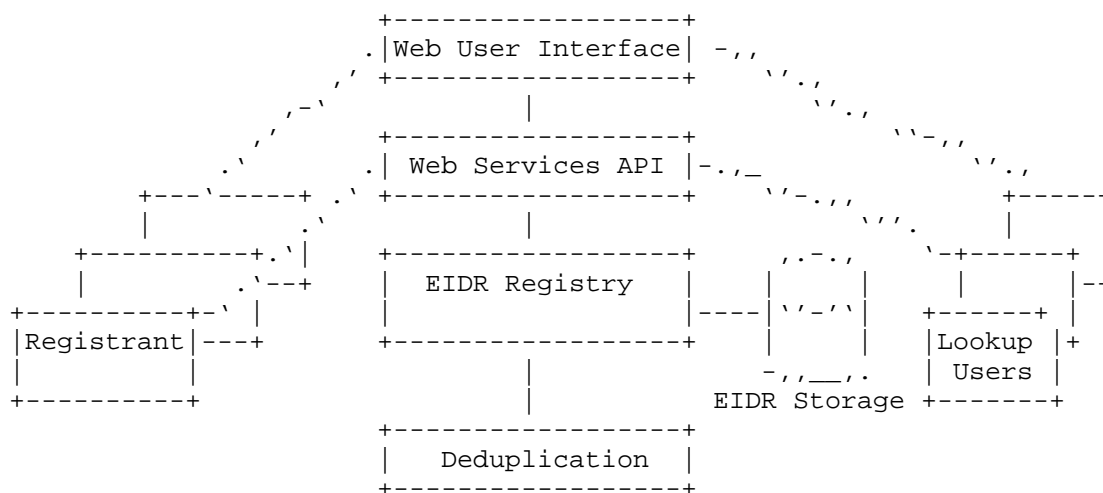


Figure 6 Entertainment Identifier Registry diagram

From the analysis of above section, the content identifier specified in EIDR is in line with the requirements of content deduplication proposed in this draft. In this document, it is suggested that we introduce content identifier format and a mechanism for generating it into CDNi to address content duplication issue.

4.4. Description of Content De-duplication

In this section the details of the solutions that use CDNi Content Naming mechanism for content de-duplication are discussed.

Content Identifier can be defined as a metadata object. The metadata object can be distributed with/without the actual content item from uCDN to dCDN for storage in the dCDN, if the uCDN wants to pre-position the content item to the dCDN before the actual user request. The content Identifier metadata object binds with the Resource Identifier to identify the storage of the content in the uCDN and forms a content identification model for the content item. By using this content identification model, an interconnected CDN is able to detect content repetition. The content status must be synchronously updated by the interconnected CDN. According to content status, the interconnected CDN can determine whether the resource copy is cached or not.

We present several procedures for optimized implementation for preventing CDNi content repetition.

4.4.1. Pre-Positioned Content Acquisition

The following flow illustrates how the two uCDNs successively pre-position the same content in the dCDN. In this flow, the content to be pre-positioned in the dCDN is identified by different Resource Identifiers corresponding to uCDN A and uCDN B.

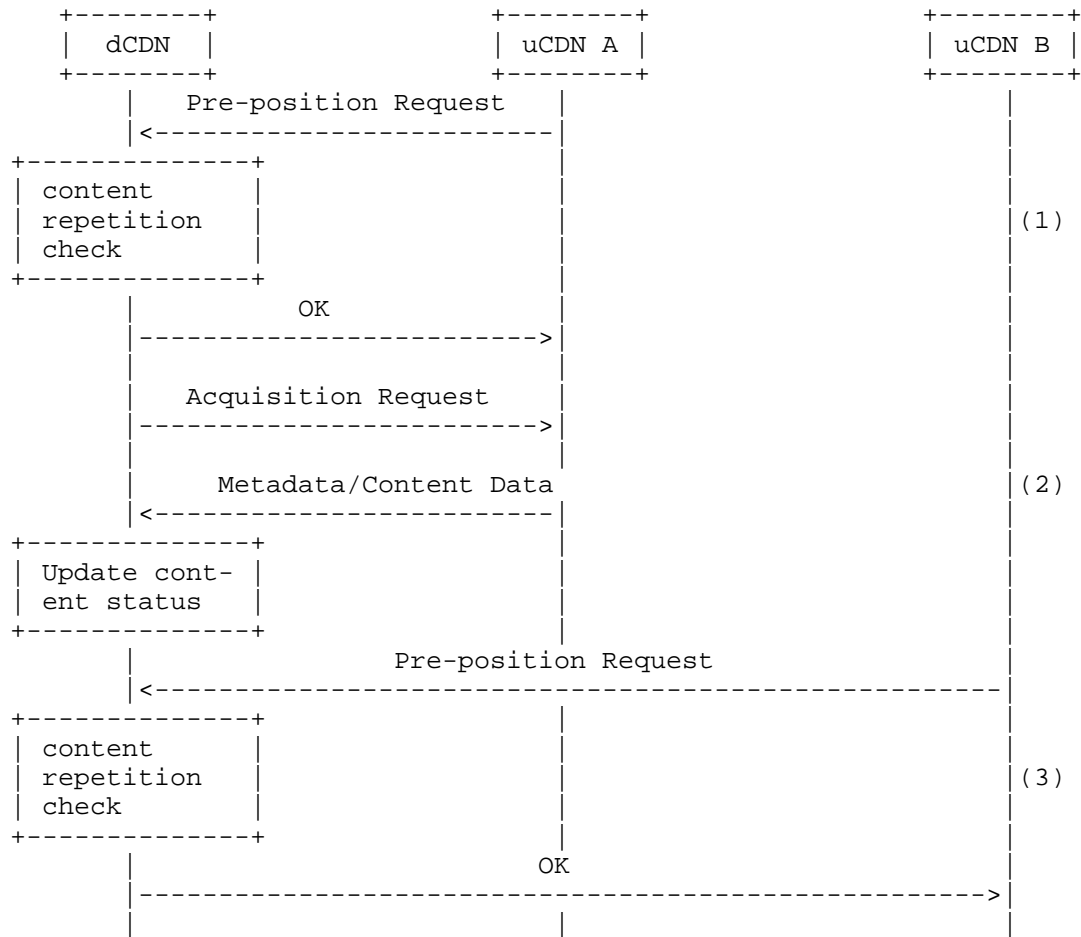


Figure 7 Acquisition of Pre-Positioned Content

The steps that are illustrated in the figure are as follows:

1. The uCDN A requests that the dCDN pre-positions a particular content item identified by its Resource Identifier and Content

Identifier. This message is sent via Trigger Interface. On reception of this message, the dCDN checks whether the same content item is cached by looking up its stored content identification model with the Content Identifier. If the metadata does not exist, the dCDN replies to uCDN A with an OK message to notify that no such copy has been cached and content pre-position is required.

2. The dCDN acquires metadata of the content or the content itself from uCDN A. Once the content is pre-positioned, dCDN updates the content status and maintains the binding between Resource Identifier and Content Identifier metadata.

3. The uCDN B requests that dCDN pre-positions the same content item identified by its Resource Identifier and Content Identifier. On reception of this message, the dCDN looks up its stored content identification model with the Content Identifier. As such metadata exists, dCDN determines that the content is already cached. Then, dCDN replies to uCDN A with an OK message to notify that the same copy has already been cached and the pre-position request should be cancelled. The dCDN locally binds the new Resource Identifier provided by uCDN B with the Content Identifier.

4.4.2. Dynamic Content Acquisition

The following flows illustrates how the dCDN performs content de-duplication in cases of a cache miss and a cache hit without content pre-positioning.

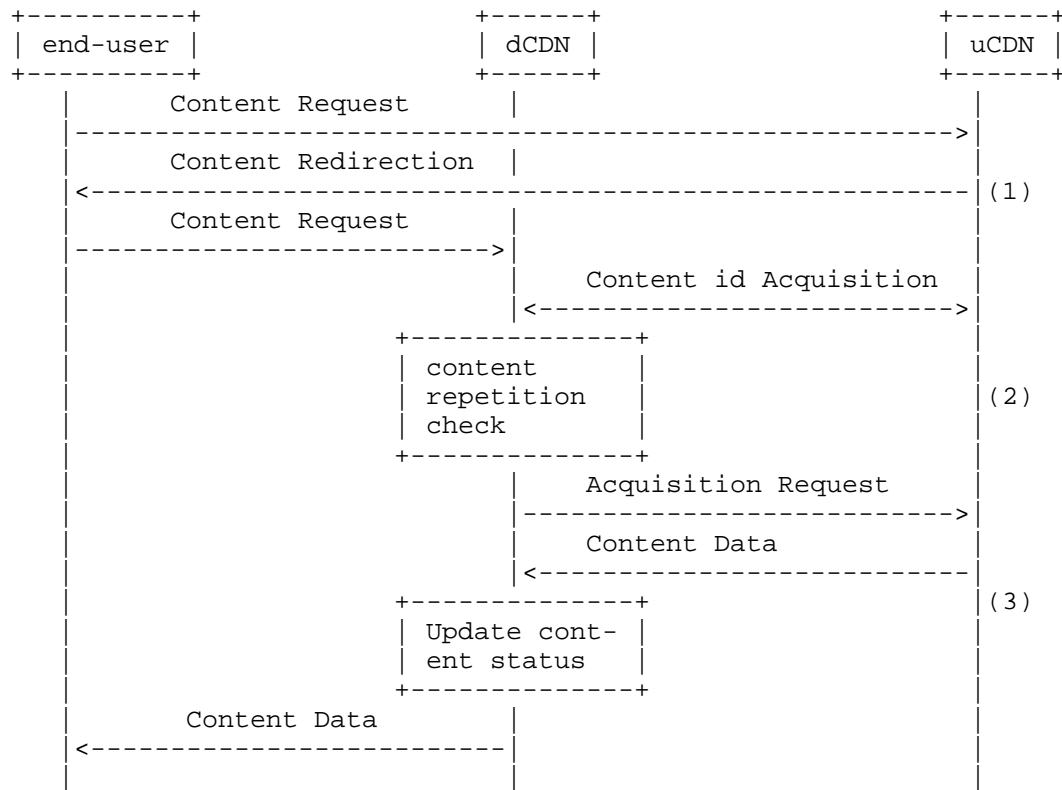


Figure 8 Dynamic Content Acquisition (cache miss case)

The steps that are illustrated in the figure are as follows:

1. A content request originated from an end-user is received by uCDN. The uCDN processes the request and recognizes that the end-user is best served by a dCDN. So uCDN redirects the request to the dCDN by sending redirection response to the end-user who then requests the content from the dCDN. The uCDN encapsulates Resource Identifier of the requested content item in the redirection response.
2. On reception of this request, the dCDN fetches the corresponding Content Identifier from the uCDN by using the Resource Identifier pointing to the requested resource. The dCDN checks whether the same content item is cached by looking up its stored content identification model with the Content Identifier. If such metadata does not exist, the case of a cache miss is determined by the dCDN, and therefore content needs to be downloaded from the uCDN before delivered to the end-user.

3. The dCDN acquires the requested content from uCDN A. Once the content is cached, dCDN updates the content status and maintains the binding relation between Resource Identifier and the Content Identifier metadata. The dCDN then delivers content data to the end-user.

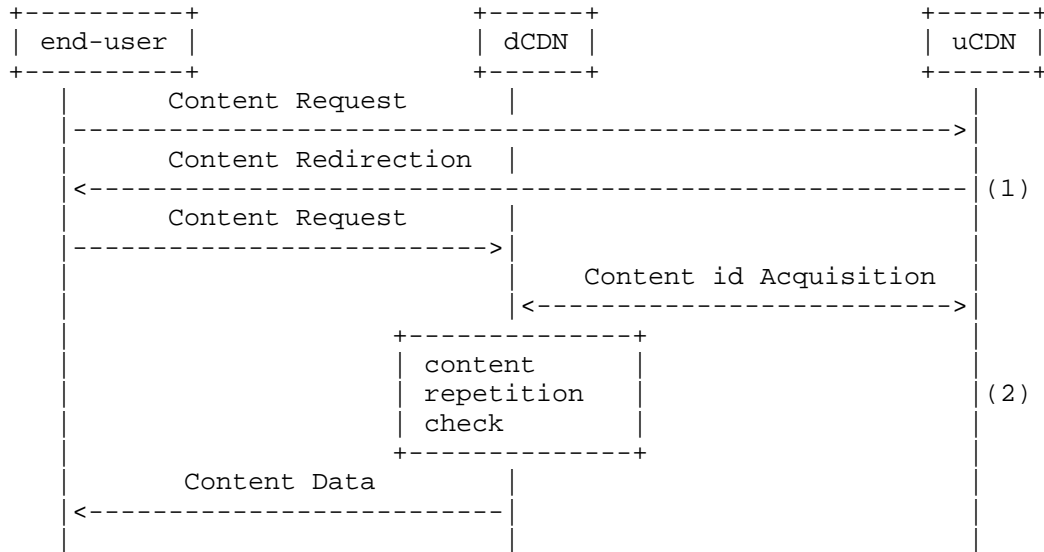


Figure 9 Dynamic Content Acquisition (cache hit case)

The steps that are illustrated in the figure are as follows:

Steps 1 and 2 are exactly the same as steps 1 and 2 of Figure 6, except that in this figure, dCDN determines the case of a cache hit according to the existence of such record in the corresponding Content Identifier metadata.

This flow differs from that in Figure 6 only in terms of not triggering dynamic content acquisition (step 3), since the content has already been cached by dCDN.

4.4.3. Content Purge and Invalidate

In general, the dCDN would assign separate location for each of its uCDN to store triggers. However, when it comes to complex CDNi deployment as discussed in this draft, dCDN is likely to receive multiple trigger operations coming from different uCDNs on the same content. If one of the uCDNs requests to invalidate certain content,

after receiving the Invalidated Trigger, dCDN will first identify the content using the Content Identifier and mark Invalid in the Trigger Status Resource corresponding to the requesting uCDN. By doing so, this content is unavailable to this uCDN before it is re-validated. The access to this content by other uCDNs will not be impacted.

If one of the uCDNs requests to purge certain content, after receiving the Purge Trigger, dCDN will identify the content using the Content Identifier and mark Invalid in the Trigger Status Resource corresponding to the requesting uCDN. By doing this, this uCDN is not able to make any further operation on this content, while the content itself is not deleted from the cache of dCDN. The access to this content by other uCDNs will not be impacted. Only when all the uCDNs binded to this content request to purge the same content and dCDN accepts all these requests will the content be purged from dCDN.

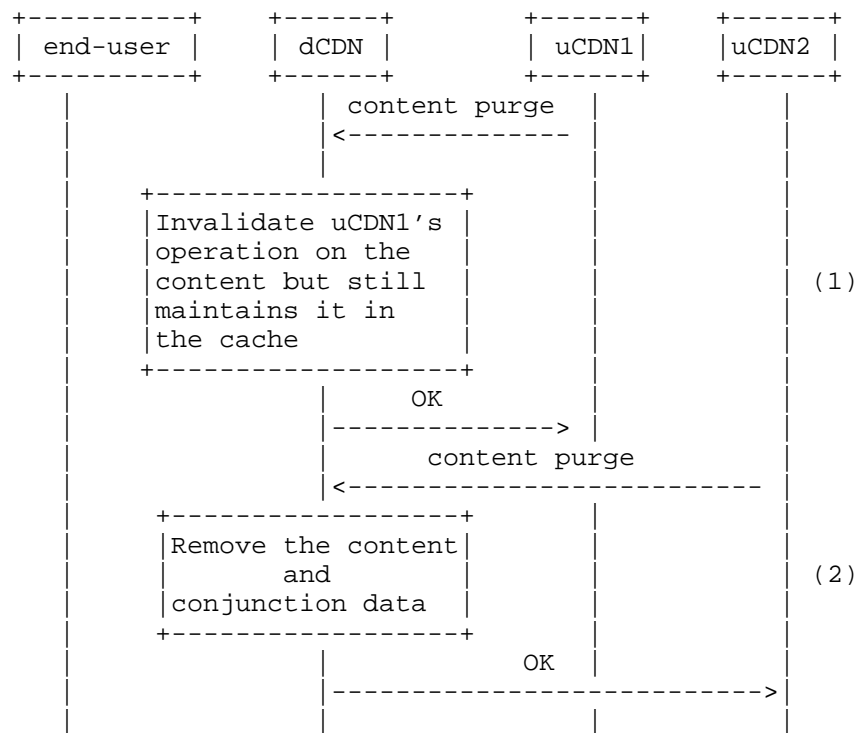


Figure 10 Removal of Content

A premise is that the content copy to be purged has already been cached in the dCDN from the uCDN. The Content Identifier of the

content is linked with two Resource IDs from uCDN1 and uCDN2 respectively. The steps illustrated in the figure are as follows:

1. The uCDN1 requests the dCDN to remove some content resource identified by the Content Identifier due to the deployment policy or expiration of content's life-time. As not only uCDN1 has been binded to this content, dCDN then only invalidates uCDN's operation on the requested content but still maintains it in the cache so that other uCDNs (e.g. uCDN2) binded to this content can still operate on this content. It then replies an OK response to uCDN1.

2. If uCDN2 also requests dCDN to remove the same content, dCDN will remove the content and all the conjunction metadata. It then replies an OK response to uCDN2.

5. Security Considerations

To be discussed/aded later.

6. IANA Considerations

This document has no IANA Considerations.

7. Acknowledgments

The authors would like to thank Francois Le Faucheur, Kevin Ma, Theodore Zahariadis, Ben Niven-Jenkins, Ram Krishnan, and Marcin Pilarski for valuable inputs, suggestions, and discussions.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. Informative References

[Data Deduplication Techniques]
Ao, L., Shu, JW., and MQ. Li, "Data Deduplication Techniques", May 2010.

[EIDR WHITEPAPER]
EIDR, EIDR., "UNIVERSAL UNIQUE IDENTIFIERS IN MOVIE AND

TELEVISION SUPPLY CHAIN MANAGEMENT", October 2010.

[I-D.ietf-cdni-framework]

Peterson, L. and B. Davie, "Framework for CDN Interconnection", February 2013.

[I-D.ietf-cdni-requirements]

Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", December 2012.

[I-D.murray-cdni-triggers]

Murray, R., Niven-Jenkins, B., and Velocix, "CDN Interconnect Triggers", March 2013.

[I-D.narten-iana-considerations-rfc2434bis]

Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", draft-narten-iana-considerations-rfc2434bis-09 (work in progress), March 2008.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

[RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", September 2012.

Authors' Addresses

WeiYi Jin
ZTE Corporation
Nanjing, 210012
China

Phone: +86 025-52871364
Email: jin.weiya@zte.com.cn

Mian Li
ZTE Corporation
Nanjing, 210012
China

Phone: +86 025-88014641
Email: li.mian@zte.com.cn

Bhumip Khasnabish
ZTE Corporation
New Jersey, 07960
USA

Phone: +001-781-752-8003
Email: bhumip.khasnabish@zteusa.com, vumipl@gmail.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 14, 2013

F. Le Faucheur
M. Viveganandhan
K. Leung
Cisco
July 13, 2012

CDNI Logging Formats for HTTP and HTTP Adaptive Streaming Deliveries
draft-lefaucheur-cdni-logging-delivery-01

Abstract

The interconnection of Content Distribution Networks (CDNs) is motivated by several use cases. CDN Interconnection can be achieved through four CDNI interfaces, one of which is the CDNI Logging interface. This document discusses CDNI logging formats for content deliveries performed using HTTP or HTTP adaptive streaming.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 14, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Terminology	4
2. CDNI Logging for regular HTTP Delivery	4
2.1. Regular HTTP Log Triggers	4
2.2. Regular HTTP Log Fields	4
3. CDNI Logging for HTTP Adaptive Streaming	6
3.1. HAS Chunk-Based Log Triggers	6
3.2. HAS Chunk-Based Log fields	6
4. Performance Monitoring	7
5. CDNI Log Encoding and Transport	7
5.1. CDNI Log Header Information	8
5.2. CDNI per-Log-Record Information	8
5.3. CDNI Log Footer Information	9
5.4. CDNI Customized Log Format	9
6. Impact on CDNI Metadata Requirements	9
7. Impact on CDNI Footprint and Capabilities Advertisement	10
8. Generation of CDNI Logs	10
9. IANA Considerations	10
10. Security Considerations	11
11. Acknowledgements	11
12. References	11
12.1. Normative References	11
12.2. Informative References	11
Authors' Addresses	12

1. Introduction

The interconnection of Content Distribution Networks (CDNs) is motivated by several use cases, such as those described in [I-D.ietf-cdni-use-cases]. The overall problem space for CDN Interconnection (CDNI) is described in [I-D.ietf-cdni-problem-statement]. The CDN Interconnection framework is defined in [I-D.ietf-cdni-framework] and the requirements for the CDN Interconnection solution are specified in [I-D.ietf-cdni-requirements].

One of the CDNI interfaces defined in these documents is the CDNI Logging interface whose description is quoted here:

"o CDNI Logging interface: This interface allows the Logging system in interconnected CDNs to communicate the relevant activity logs in order to allow log consuming applications to operate in a multi-CDN environments. For example, an upstream CDN may collect delivery logs from a downstream CDN in order to perform consolidated charging of the CSP or for settlement purposes across CDNs. Similarly, an upstream CDN may collect delivery logs from a downstream CDN in order to provide consolidated reporting and monitoring to the CSP."

The logging interface is discussed in details in [I-D.bertrand-cdni-logging]. The present document identifies a number of additional considerations regarding logging for content delivery performed using HTTP.

[I-D.brandenburg-cdni-has] discusses the interactions of HTTP Adaptive Streaming with CDNI Logging and provides recommendations on how to specifically perform CDNI Logging for delivery performed using HTTP Adaptive Streaming. The present document discusses in more details how these recommendations would impact the CDNI Logging interface.

Section 2 discusses the aspect of CDNI Logging that are specific to content delivery performed via regular HTTP. Section 3 discusses the aspect of CDNI Logging that are specific to content delivery performed via HTTP adaptive streaming. Section 5 discusses the aspect of CDNI Logging that are generic to all the CDNI Logs for delivery.

CDNI Logging for other events than content delivery (e.g. failures, request routing, service monitoring,...) are not discussed in the present document.

1.1. Terminology

This document uses all terminology defined in [I-D.brandenburg-cdni-has].

2. CDNI Logging for regular HTTP Delivery

This section discusses the triggers on which a CDNI log is generated for delivery of content using regular HTTP delivery, as well as the fields contained in the corresponding log.

2.1. Regular HTTP Log Triggers

A CDNI Log is generated for a regular HTTP delivery on the following triggers:

Event	Description
content Request	Reception and processing of a request for a content

2.2. Regular HTTP Log Fields

A CDNI Log for regular HTTP delivery contains the following fields (or a subset thereof as discussed in Section 5.4):

Field	Description	Examples
Current-Time	Time, in milliseconds, at which the request was received.	[20/Feb/2012:00:29.510+0200]
Time-to-Serve	Time, in microseconds, taken to complete the request.	952195

Client-IP	IP address of the requesting client.	203.0.113.2
Action	Squid action describing how the request was treated locally (e.g., cache hit/miss)	TCP_HIT, TCP_MISS, ...
Status-Returned	HTTP response code.	200, 404, ...
Bytes-Transferred	Bytes sent to the client, including the headers.	23567992
Method	HTTP request method.	GET
URI	URI of requested content.	http://cache3.cdn1.com/movie/ice/icemovie.mpg
Content-Type	MIME-Type from the reply header.	video/mpeg
User-Agent (and possibly some other HTTP headers)	content of the User-Agent HTTP Header	Mozilla/5.0 (X11; Linux x86_64; rv:12.0) Gecko/20100101 Firefox/12.0

URI-Signing-Validation	Flag indicating whether URI signature validation was performed	0/1
------------------------	--	-----

3. CDNI Logging for HTTP Adaptive Streaming

This section discusses the CDNI logs for delivery of content using HTTP adaptive streaming. In line with the recommendations of [I-D.brandenburg-cdni-has], this document proposes that a log record be generated for delivery of each chunk or manifest file.

3.1. HAS Chunk-Based Log Triggers

A chunk-based Log record is generated for HAS delivery on the following triggers:

Event	Description
Chunk/Manifest Request	Reception of a request for a Segment (or Manifest File)

3.2. HAS Chunk-Based Log fields

A chunk-based Log record for HTTP adaptive streaming may contain all the same fields as a CDNI Log for a regular HTTP request, as well as the following additional fields:

Field	Description	Examples
Content Collection ID	Identifier for Content Collection	format and scope of unicity are TBD

Session-id	a string generated by the delivering CDN and unique (to the delivering CDN) to identify the Session. (*)	6141F5795BE774691D234A0465B9667A
------------	--	----------------------------------

(*) The Session-ID value to be included in a log record by the delivering CDN is such that:

- o different per-chunk log records with the same Session-ID value must correspond to the same user session (.i.e delivery of same content to same enduser at a given point in time)
- o log records for different chunks of the same user session (.i.e delivery of same content to same enduser at a given point in time) should be provided with the same session-ID value. While undesirable, there may be situations where the delivering CDN uses more than one session-ID value for different per-chunk log records of a given session, for example in scenarios of fail-over or load-balancing across multiple Surrogates and where the delivering CDN does not implement mechanism to synchronize session-IDs across Surrogates.

4. Performance Monitoring

The CDNI Log fields listed in the previous sections allow monitoring of essential delivery performance indicators across the CDN Mesh. For example, for a regular HTTP delivery, these fields allow tracking of the time taken to serve the requested content, tracking of content delivery failures, tracking of partial deliveries and tracking of cache hit ratios. As another example, for HTTP adaptive streaming, these fields allow tracking of the presentation (and its fluctuation over time) served to the End-User as well as the End-User random content access (e.g. Play/Stop/Seek).

Subsequent versions of this document may discuss potential additional log fields for enhanced performance monitoring.

5. CDNI Log Encoding and Transport

Details for CDNI Log encoding and transport will be specified in subsequent versions. We observe that this is expected to allow optional use of common compression techniques (e.g. gzip). However,

Section 5.1 identifies the information that is to be included in the header of CDNI Logs, Section 5.2 identifies information that is to be attached to every CDNI Log record and Section 5.3 identifies the information that is to be included in the footer of CDNI Logs. Finally, Section 5.4 discusses the notion of customized Logging.

5.1. CDNI Log Header Information

The header of CDNI Logs contains the following fields:

Field	Description	Examples
Format-Version	Version of the CDNI Log format.	v1.0
Log-Field-List	The list of the fields provided in the log records	time cs-method cs-uri
Log-ID	Unique identifier for the CDNI Log (facilitates detection of duplicate Logs and tracking in case of aggregation).	
Log-Timestamp	Time, in milliseconds, the CDNI Log was generated.	[20/Feb/2012:00:29.510+0200]

5.2. CDNI per-Log-Record Information

In addition to the log fields discussed in previous sections, each CDNI log record contains the following fields:

Field	Description	Examples
Log Record Digest	Digest of the Log records (facilitates recovery of uncorrupted Log records inside a corrupted CDNI Log)	

5.3. CDNI Log Footer Information

The footer of CDNI Logs contains the following fields:

Field	Description	Examples
Log Digest	Digest of the complete Log (facilitates detection of Log corruption)	

5.4. CDNI Customized Log Format

This document proposes that customized logs be supported by CDNI in the following manner:

- o the uCDN uses the CDNI Metadata interface to indicate to the dCDN which subset of the CDNI logging fields are to be provided in a log record for corresponding to a request for a given content
- o the dCDN provides, via the CDNI Logging interface, log records containing the subset of CDNI logging fields requested by the uCDN.
- o The dCDN explicitly lists in the CDNI Log Header the fields actually provided (as discussed in Section 5.1).

6. Impact on CDNI Metadata Requirements

We request that the following requirements be added in section 6 of [I-D.ietf-cdni-requirements]:

"

META-X [HIGH] The CDNI Metadata Distribution interface shall support an OPTIONAL mechanism allowing the Upstream CDN to indicate to the Downstream CDN which CDNI Log fields are to be provided for all, for specific sets of, or for specific content items delivered using HTTP. A CDNI implementation that does not support this optional CDNI Metadata Distribution Interface mechanism MUST ignore this log format indication and generate CDNI logging format for adaptive streaming using the default set of CDNI Logging fields. [Editor's note: A default set of logging fields need to be defined]

META-X [MID] The CDNI Metadata Distribution interface shall allow the uCDN to signal to the dCDN the Content Collection ID value for all, for specific sets of, or for specific content items delivered using

HTTP. Whenever the dCDN is instructed by the uCDN (using the customized logging mechanism described in Section 5.4) to report the Content Collection ID field in the log records, the dCDN is to use the value provided through the CDNI Metadata interface for the corresponding content.

"

7. Impact on CDNI Footprint and Capabilities Advertisement

We request that the following requirement be added in section 5 of [I-D.ietf-cdni-requirements]:

"

REQ-X [MID] The CDNI Request Routing/Footprint and Advertisement Interface shall support advertisement of the following capabilities:

- o support for customized CDNI Logging
- o support of Content Collection ID logging
- o support for Session-ID logging

"

8. Generation of CDNI Logs

Like other CDNI interfaces, the CDNI Logging interface specifies operations across CDNs and not inside a CDN. Therefore, the log formats specified in this document apply to CDNI logging information exchanged across CDNs and does not constrain the process for generating such inter-CDN logs within a given CDN. The format of the logs generated by a given CDN Surrogate is beyond the scope of the present document. We observe that a given CDN could elect to have its Surrogates natively generate logs in the same format as the one to be used for exchange with another CDN, or that the CDN could elect to have its Surrogates generate logs in any other format (as long as they include the necessary information) and have these logs then reformatted prior to exchange with another CDN.

9. IANA Considerations

[This will be specified in subsequent versions].

10. Security Considerations

CDNI Logs exchanged over the CDNI Logging interface can be consumed by very sensitive applications including inter-CDN accounting and billing. The associated security concerns are discussed in [I-D.ietf-cdni-framework]. At this stage, we observe that the CDNI Logging interface can leverage the existing security mechanisms supported by the underlying transport protocol that will be selected for transport of CDNI Logs (e.g. to support authentication of the entities exchanging CDNI Logs through the CDNI Logging interface, to support privacy and protection against spoofing). This will be further discussed in subsequent versions of this document.

11. Acknowledgements

The authors want to thank Gene Halbrooks for his input into this document.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

12.2. Informative References

[I-D.bertrand-cdni-logging]

Bertrand, G. and S. Emile, "CDNI Logging Interface", draft-bertrand-cdni-logging-00 (work in progress), February 2012.

[I-D.brandenburg-cdni-has]

Brandenburg, R., Deventer, O., Faucheur, F., and K. Leung, "Models for adaptive-streaming-aware CDN Interconnection", draft-brandenburg-cdni-has-03 (work in progress), July 2012.

[I-D.ietf-cdni-framework]

Peterson, L. and B. Davie, "Framework for CDN Interconnection", draft-ietf-cdni-framework-00 (work in progress), April 2012.

[I-D.ietf-cdni-problem-statement]

Niven-Jenkins, B., Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem

Statement", draft-ietf-cdni-problem-statement-08 (work in progress), June 2012.

[I-D.ietf-cdni-requirements]

Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", draft-ietf-cdni-requirements-03 (work in progress), June 2012.

[I-D.ietf-cdni-use-cases]

Bertrand, G., Emile, S., Burbridge, T., Eardley, P., Ma, K., and G. Watson, "Use Cases for Content Delivery Network Interconnection", draft-ietf-cdni-use-cases-09 (work in progress), July 2012.

Authors' Addresses

Francois Le Faucheur
Cisco Systems
Greenside, 400 Avenue de Roumanille
Sophia Antipolis 06410
France

Phone: +33 4 97 23 26 19
Email: flefauch@cisco.com

Mahesh Viveganandhan
Cisco Systems
375 East Tasman Drive
San Jose 95134
USA

Email: mvittal@cisco.com

Kent Leung
Cisco Systems
3625 Cisco Way
San Jose 95134
USA

Phone: +1 408 526 5030
Email: kleung@cisco.com

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: January 10, 2013

Y. Le Louedec
A. Marrec
G. Bertrand
France Telecom - Orange
M. Pilarski
Orange Polska / WUT
July 9, 2012

CDNI Request Routing
draft-lelouedec-cdni-request-routing-00

Abstract

The present document proposes to clarify the CDNI Request Routing interface introduced in [I-D.ietf-cdni-framework] and [I-D.ietf-cdni-problem-statement], as well as related terminology.

In particular the present document proposes to split the CDNI Request Routing interface into two separate interfaces with clearer roles, named respectively CDNI Routing interface and CDNI Downstream Resource Identifier Signaling interface (CDNI DRIS interface).

This part of the CDN interconnection framework the IETF has been referring to so far with the term "CDNI Request Routing" is just another routing, signaling and forwarding problem in a long series of the telecommunication history. For example, one can draw a direct analogy between the IP/MPLS-TE framework and the CDN interconnection framework.

In addition, this document recommends that the specification of ALL CDN interconnection interfaces in the scope of the CDNI IETF WG relies on the equivalent concept to IP prefix for CDN interconnection, named 'contentRequestScope'. This highly useful and powerful concept SHALL be used to simplify the specification of ALL CDN interconnection interfaces, as well as to ensure performance and scalability in CDN interconnection.

All these proposals can be smoothly integrated in the WG drafts, especially [I-D.ietf-cdni-framework] and [I-D.ietf-cdni-requirements], as they essentially propose (useful) clarifications of the existing framework.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 10, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
2. Motivations and Terminology Clarifications	4
2.1. The CDNI Request Routing interface should be split	5
2.2. The routing function of the CDNI Request Routing interface should be clarified	6
2.3. CDNI request redirection strategies	7
3. Overview of the CDNI Routing interface	8
4. The CDNI Downstream Resource Identifier Signaling (DRIS) Interface	11
4.1. Overview of the CDNI DRIS interface	11
4.2. Overview of CDNI DRIS operations	15
4.2.1. Case of iterative CDNI request redirection	16
4.2.2. Case of recursive CDNI request redirection	18
4.3. Key points to note about the CDNI DRIS interface	23
5. CDNI request routing is just another routing and signaling problem	24
6. Conclusion and recommendations	26
7. Acknowledgments	27
8. IANA Considerations	28
9. Security Considerations	28
10. Informative References	28
Appendix A. Proposal for Section 4.2 of CDNI framework draft	28
Authors' Addresses	29

1. Introduction

The present document proposes to clarify the CDNI Request Routing interface introduced in [I-D.ietf-cdni-framework] and [I-D.ietf-cdni-problem-statement], as well as the related terminology.

In particular the present document proposes to split the CDNI Request Routing interface into two separate interfaces with clearer roles, named respectively CDNI Routing interface and CDNI Downstream Resource Identifier Signaling interface (CDNI DRIS interface).

Section 2 presents the motivations for splitting the CDNI Request Routing interface. It also proposes to review and to expand the terminology about the terms iterative request routing and recursive request routing.

Section 3 and Section 4 provide a short overview respectively of the CDNI Routing interface and of the CDNI DRIS interface. The detailed specifications of each of these interfaces will be made public shortly by the authors in dedicated IETF drafts.

Section 5 provides an analogy with IP/MPLS-TE framework. This just aims at helping readers to understand the proposals and recommendations in this document.

Section 6 concludes this draft with recommendations for the IETF CDNI WG. This includes recommendations about the CDNI WG charter, as well as about [I-D.ietf-cdni-framework] and [I-D.ietf-cdni-requirements].

Appendix A proposes a text to replace Section 4.2 of [I-D.ietf-cdni-framework].

2. Motivations and Terminology Clarifications

Section 2.1 and Section 2.2 present the motivations for splitting and clarifying the CDNI Request Routing interface.

This document adopts the terminology described in [I-D.ietf-cdni-problem-statement] and [I-D.ietf-cdni-framework], except for the terms 'iterative request routing' and 'recursive request routing'. Section 2.3 proposes to review and to expand the terminology about these two terms.

In addition this document extends this terminology with the terms 'CDNI routing interface', 'CDNI DRIS interface' and 'contentRequestScope' introduced and illustrated respectively in Sections 3, 4, and 5. It also introduces the term 'delivery

resource'. From the perspective of the uCDN, a delivery resource is a delivery node or a request routing system towards which a content request may be redirected: either a delivery node inside the uCDN, or the request routing system of a CDN different from the uCDN, or a delivery node within a CDN different from the upstream CDN.

2.1. The CDNI Request Routing interface should be split

One key message of the present document is that the term "CDNI Request Routing" is confusing as it makes a mix-up of two clearly different sets of processes and interfaces.

As described in [I-D.ietf-cdni-problem-statement], the CDNI Request Routing interface's role is twofold:

1. "To enable a downstream CDN to provide to the upstream CDN (static or dynamic) information (e.g., resources, footprint, load) to facilitate selection of the downstream CDN by the upstream CDN request routing system when processing subsequent content requests from User Agents."
2. "To allow the downstream CDN to control what the upstream Request Routing function should return to the User Agent in the redirection message."

These two functions are fundamentally different. Besides, depending on the considered use cases, described in [I-D.ietf-cdni-use-cases], they could be implemented differently (e.g., the first function could be implemented manually while the second function would be implemented with a network protocol, or vice versa, or they could rely on different network protocols, etc.).

This idea that the request routing interface comprises two parts has started to be visible in the latest version of the Section 4.2 in [I-D.ietf-cdni-framework]. We think that it is a good first step that requires clarifications. Indeed, [I-D.ietf-cdni-framework] mentions a 'synchronous part' that makes again a mix-up between operations involving communications with the user agent and operations only between the CDNs. And the latter ones can be purely asynchronous operations in some cases, as illustrated in Section 4 of the present memo.

Therefore, in order to simplify and accelerate the specification of the CDNI interfaces, the present document proposes to split the current CDNI Request Routing interface into two separate interfaces with clearer roles, further detailed in Section 3 and Section 4 respectively:

1. The CDNI Routing Interface
 2. The CDNI Downstream Resource Identifier Signaling (DRIS) Interface.
- 2.2. The routing function of the CDNI Request Routing interface should be clarified

Quoting [I-D.ietf-cdni-problem-statement], "The CDNI Request Routing interface enables a Request Routing function in an upstream CDN to query a Request Routing function in a downstream CDN to determine if the downstream CDN is able (and willing) to accept the delegated content request".

The "ability" and the "willingness" of the downstream CDN to accept the delegated content request match two fully different processes in CDN interconnection. And the description of both these processes should be reviewed and clarified for the following reasons:

- o Regarding the former process, i.e. related to the "ability" ("enables a Request Routing function in an upstream CDN to query a Request Routing function in a downstream CDN to determine if the downstream CDN is able (...) to accept the delegated content request"), a routing protocol is used to achieve such a process, called routing process, in many other technologies and networks (IP, ATM, etc.). In all these technologies, it is the downstream entity that provides routing information to the upstream entity. The same approach should be applied to CDN interconnection. The reverse approach ("uCDN queries it downstream CDNs dCDN 1, dCDN 2, dCDN 3, and then it selects one of them based on their responses") would suffer from latency, signaling overhead and/or lack of reactivity to events that impact the ability of the downstream CDNs to accept delegated content requests.
- o Regarding the latter process, i.e. related to the "willingness" ("enables a Request Routing function in an upstream CDN to query a Request Routing function in a downstream CDN to determine if the downstream CDN (...) is willing (...)to accept the delegated content request"), it is to be noted that the relationship between a uCDN and a dCDN is always in the frame of a contractual agreement between the administrative entity owning the uCDN, acting as the customer, and the administrative entity owning the dCDN, acting as the service provider. Therefore, consider a case where
 - * the uCDN has a content request to redirect which is in full conformance with the terms of this contractual agreement, and

- * the uCDN has selected this dCDN.

As the uCDN knows, thanks to the routing information exchanged via the aforementioned routing process, that the dCDN is able to accept this content request, the uCDN MAY redirect the content request to the dCDN and the dCDN MUST accept the content request. Exchanging over the CDNI Request Routing interface information about the "willingness" of the dCDN to accept the content request is not relevant.

2.3. CDNI request redirection strategies

The terms "Iterative CDNI request routing" and "Recursive CDNI request routing" are defined in [I-D.ietf-cdni-framework]. As explained above the term "CDNI request routing" is confusing, therefore, the present document proposes to use the terms "Iterative CDNI request redirection" and "Recursive CDNI request redirection" instead. We consider that these new terms are more appropriate than the definitions provided in [I-D.ietf-cdni-framework].

The definition of "Recursive CDNI request routing" given in [I-D.ietf-cdni-framework] indicates "...that the Downstream CDN may elect to have the request redirected directly to a delivery node inside the Downstream CDN, to the Request-Routing System of the Downstream CDN, to another CDN, or to any other system that the Downstream CDN sees as fit for handling the redirected request."

In order to clarify this point, and for the purpose of Section 3 to Section 6, the present document introduces the terms "full-recursive CDNI request redirection" and "semi-recursive CDNI request redirection".

Full-recursive CDNI request redirection and semi-recursive CDNI request redirection are two subtypes of recursive CDNI request redirection. A recursive CDNI request redirection is either a full-recursive CDNI request redirection or a semi-recursive request redirection.

"Full-recursive CDNI request redirection" refers to the case where the considered CDN redirects the content request directly to a delivery node of another CDN, as illustrated in Figure 4. It means the content request is redirected:

- o either directly to a delivery node inside the Downstream CDN,
- o or directly to a delivery node inside a downstream CDN of the downstream CDN,

- o or directly to a delivery node inside a downstream CDN of a downstream CDN of the downstream CDN,
- o etc. (anyway directly to a delivery node).

In contrast, "semi-recursive CDNI request redirection" refers to the case, illustrated on Figure 5, where the considered CDN does not directly redirect the content request to a delivery node. It means the request is redirected:

- o either to the request routing system of the Downstream CDN that will deliver the content
- o or to the request routing system of another CDN that will deliver the content and that is a downstream CDN of the downstream CDN,
- o or to the request routing system of another CDN that will deliver the content and that is a downstream CDN of a downstream CDN of the downstream CDN,
- o etc. (anyway not directly to a delivery node).

Full-recursive redirection has the advantage over semi-recursive redirection of being more transparent from the user agent's perspective, but the disadvantage of the downstream CDN exposing more of its internal structure (in particular, the addresses of delivery nodes) to the upstream CDN (and possibly to the upstream CDN(s) of its upstream CDN, etc.). By contrast, semi-recursive redirection does not require dCDN to expose the addresses of its delivery nodes to uCDN. The specifications for the CDNI interfaces elaborated within the IETF CDNI WG MUST allow to support both these recursive CDNI request redirection strategies.

3. Overview of the CDNI Routing interface

The CDNI routing interface is dedicated to the CDNI routing process. The CDNI routing process consists in the advertisement of CDNI routing information from the downstream CDN to the upstream CDN.

Note: The upstream CDN never provides the downstream CDN with any CDNI routing information.

CDNI routing information details "capabilities" of the downstream CDN that the upstream CDN must take into account in its CDN selection process. The CDN selection process consists for the upstream CDN to select the CDN (either the upstream CDN or one of its downstream CDN(s)) that should handle the content request the upstream CDN

receives from the user agent.

Note: The term "capabilities" refer to the capacity of the downstream CDN to handle correctly content requests from user agents. It does not necessarily mean that the downstream CDN will satisfy (alone) all these content requests by delivering the requested content to the user agent. Indeed the downstream CDN may have its own downstream CDN(s) and may decide that some or all of these content requests will be handled by its own downstream CDN(s).

Figure 1 provides a short illustration of the CDNI routing interface. The upstream CDN on Figure 1 has two downstream CDNs: CDN1 and CDN2.

Via their respective CDNI routing interfaces established with the upstream CDN, CDN1 and CDN2 provide the upstream CDN with CDNI routing information about the capabilities that they offer to the upstream CDN in the frame of their respective CDNI agreements contracted with the upstream CDN.

The inner architecture of the upstream CDN is beyond the scope of the current charter of the IETF CDNI WG. Figure 1 provides a high level representation of a possible implementation just in order to ease understanding of the global picture.

Basically and typically, the CDNI routing information received from CDN1 and CDN2 will feed a CDNI routing table controlled by a routing module in the upstream CDN. And every time the upstream CDN receives a content request from a user agent (either a DNS request in case of DNS based request redirection, or an HTTP request in case of HTTP based request redirection, etc.), its CDN selection module (in charge of the CDN selection process) will consult this CDNI routing module to decide about which CDN will handle this request (the upstream CDN, CDN1, or CDN2). In the example on Figure 1, the upstream CDN selects CDN 2 for the considered content request.

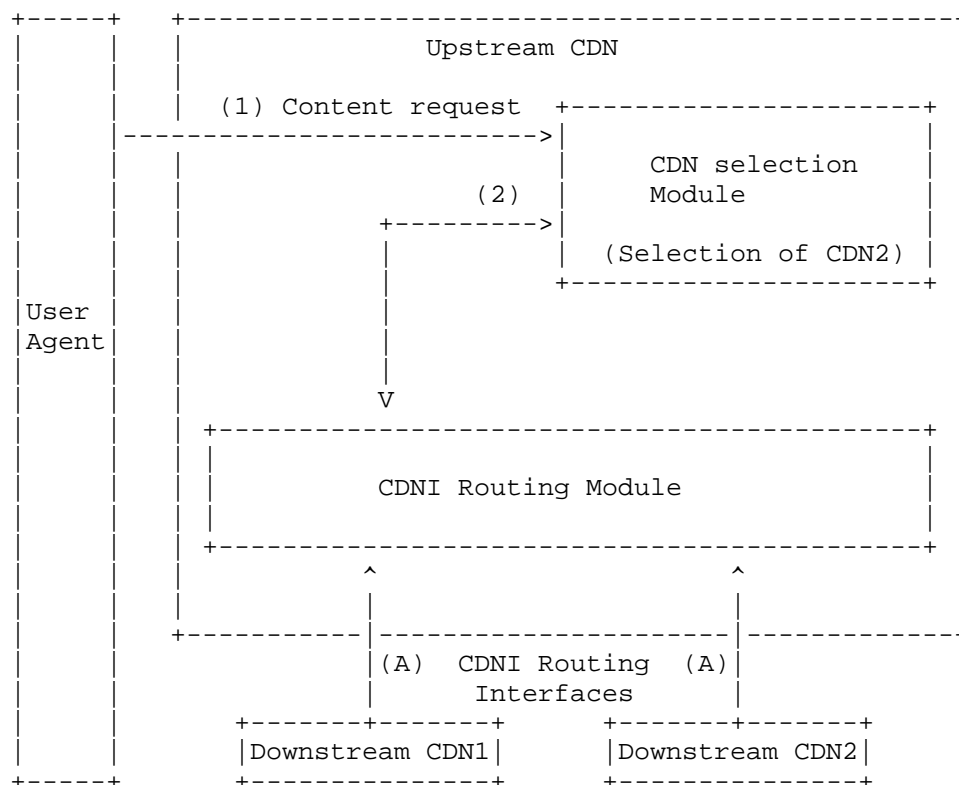


Figure 1: CDNI routing interface

The operations shown on Figure 1 are as follows:

1. A content request from a user agent (and/or from an intermediate local DNS in case it is a DNS request) arrives at the upstream CDN.
 2. Within the upstream CDN, the CDN selection module consults the routing module to decide which CDN will handle this content request (i.e. whether it will be the upstream CDN or one of its downstream CDNs). The interface between these internal modules is out of standardization scope.
- A. The operations achieved over the CDNI Routing interfaces aim at providing all information about the capabilities offered by the downstream CDNs to the upstream CDN in the frame of their respective CDNI agreements contracted with the upstream CDN.

The operations referenced with numeric indexes on Figure 1 ("1", "2") show a time dependency: every time the upstream CDN receives a content request ("1") its CDN selection process consults the routing table ("2").

In contrast, Routing information is to be exchanged continuously, with keep alive and update messages. There is no time dependency between the content requests received by the upstream CDN and the operations over the CDNI routing interfaces. By reference to the terminology used in [I-D.ietf-cdni-framework], CDNI Routing operations are asynchronous. The CDNI routing interfaces are referenced with an alphabetic index ("A") on Figure 1 to reflect this point.

4. The CDNI Downstream Resource Identifier Signaling (DRIS) Interface

4.1. Overview of the CDNI DRIS interface

AFTER the CDN selection process is achieved by the upstream CDN for a given content request (see Section 3), the upstream CDN MUST generate a response redirecting that content request towards the selected delivery resource by using in this response an identifier of that selected delivery resource.

The selected delivery resource can be:

1. Either a delivery node within the upstream CDN

This corresponds to the case where the CDN selection process of the upstream CDN selects the upstream CDN for that content request. In this case, the upstream CDN delivers the requested content to the client through one of its delivery nodes. The selected delivery resource is the upstream CDN's delivery node.

2. Or the request routing system of a CDN different from the upstream CDN

This corresponds to the case where the upstream CDN did select a downstream CDN for that content request, and where the CDNI request redirection strategy between the upstream CDN and this downstream CDN is based either on the iterative mode or on the semi-recursive mode. If the CDNI request redirection strategy between the uCDN and the dCDN is based on the iterative mode, the CDN to which the request routing system belongs is the downstream CDN selected by the upstream CDN for that content request (see Figure 3: Iterative CDNI request redirection). If the CDNI request redirection strategy between the uCDN and the dCDN is based on the semi-recursive mode, the CDN to

which that request routing system belongs can possibly be

- o the downstream CDN selected by the upstream CDN for that content request, or
- o a downstream CDN of that downstream CDN, or
- o a downstream CDN of a downstream CDN of that downstream CDN,
- o etc.,

due to the recursive nature of this mode (see Figure 5).

3. Or a delivery node within a CDN different from the upstream CDN

This corresponds to the case where the upstream CDN did select a downstream CDN for that content request, and where the CDNI request redirection strategy between the upstream CDN and this downstream CDN is based on the full-recursive mode.

The CDN to which the selected delivery node belongs can possibly be a delivery node within the downstream CDN selected by the upstream CDN for that content request, or a delivery node within a downstream CDN of that downstream CDN, or a delivery node within a downstream CDN of a downstream CDN of that downstream CDN, etc., due to the recursive nature of this mode (see Figure 4: Full-recursive CDNI request redirection).

In the cases "2." and "3.", the downstream CDN MUST provide the identifier of the selected delivery resource to the upstream CDN. The downstream CDN uses the CDNI DRIS interface to provide the upstream CDN with this identifier called "Downstream Resource Identifier" (DRI).

Note. In the recursive modes (full-recursive and semi-recursive) the upstream CDN does not need to know if the selected delivery resource belongs to the downstream CDN that the uCDN selected for that content request or to another CDN (i.e. to one downstream CDN of this downstream CDN, or to one downstream CDN of one downstream CDN of this downstream CDN, etc.). The upstream CDN just needs to get from the selected downstream CDN the "Downstream Resource Identifier" (DRI) of the selected delivery resource, so as to generate with this identifier a response redirecting that content request towards that selected delivery resource.

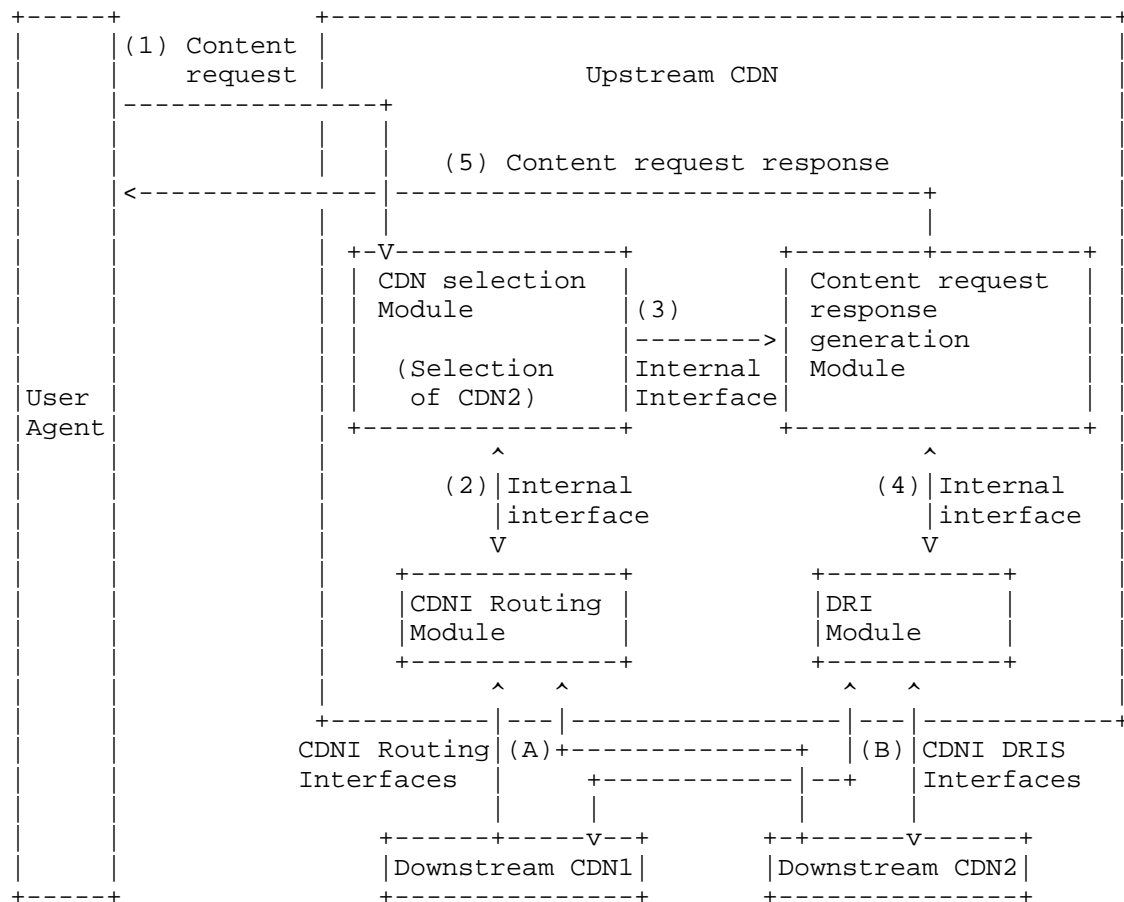


Figure 2: CDNI DRIS interface

Figure 2 is an extension of Figure 1. Again, the inner architecture of the CDN is beyond the scope of the IETF CDNI WG charter; Figure 2 provides a high level representation of a possible implementation just in order to ease the understanding of the global picture.

The operations shown on Figure 2 are as follows:

1. A content request from a user agent (and/or from an intermediate local DNS in case it is a DNS request) arrives at the upstream CDN.
2. Within the upstream CDN, the CDN selection Module consults the routing module to decide which CDN will handle this content

request (i.e. whether it will be the upstream CDN or one of its downstream CDNs). The interface between these internal modules is out of standardization scope. Let us consider that the CDN selection module selected the downstream CDN2 for that content request (so as to continue with the illustrating example introduced in Section 3).

3. Once this CDN selection is achieved, the module in charge of generating the response to the content request ('Content request response generation Module' on Figure 2) is called. The interface between these internal modules is out of standardization scope. In order to generate the response, the Content request response generation module needs to get the identifier of the selected delivery resource (DRI) to where the content request must be redirected.
 4. The Content request response generation module consults the DRI module to get this adequate DRI. The interface between these internal modules is out of standardization scope. The DRI module is in charge of getting and storing the DRI(s) provided by each downstream CDN over their CDNI DRIS interfaces with the upstream CDN.
 5. Once the Content request response generation module has this adequate DRI, it generates the response to the content request so as to the request be redirected towards the selected delivery resource.
- A. The operations achieved over the CDNI Routing interfaces aim at providing all information about the capabilities offered by the downstream CDNs to the upstream CDN in the frame of their respective CDNI agreements contracted with the upstream CDN.
- B. The operations achieved over the CDNI DRIS Interface with the downstream CDN2 aim at providing the upstream CDN with the DRI, i.e. the identifier of the selected delivery resource.

The operations referenced with numeric indexes on Figure 2 ("1", "2", "3", "4", "5") show a strict time dependency. In particular, it is to be highlighted that the CDN selection process of the upstream CDN does not take the DRI information into account. This identifier is used by the content request response generation module only after the CDN selection process is achieved.

As explained in Section 3, the CDNI routing interfaces are referenced with an alphabetic index ("A") to indicate that the CDNI routing information are exchanged asynchronously and continuously, with keep alive and update messages.

The CDNI DRIS interfaces are also referenced with an alphabetic index ("B") to indicate there is not always a systematic time dependency between the content requests received by the upstream CDN and the operations occurring over the CDNI DRIS interfaces. This depends, among others, on the CDNI request redirection strategy enforced between the CDNs (recursive or iterative). This means that the specification of the CDNI DRIS interface MUST enable synchronous and asynchronous operations, as illustrated in the next Section 4.2.

4.2. Overview of CDNI DRIS operations

This section gives an overview of operations over the CDNI DRIS interfaces for each CDNI request redirection strategy (iterative, full-recursive and semi-recursive).

All the figures present the same network topology with three cascaded CDNs so as to show very clearly the difference between the full-recursive and semi-recursive modes. The case where only two CDNs are involved can be straightforwardly deduced. In addition, the CDNI request redirection strategy applied between the first CDN and the second CDN (e.g., iterative) could differ from the strategy applied between the second CDN and the third CDN (e.g., semi-recursive). The three figures are for illustration purpose only; they do not aim at reflecting exhaustively the full flexibility that the CDN service providers have when choosing their CDNI request redirection strategies.

The same situation and content request are considered in all the three figures:

- o There is one CDNI agreement and one CDN interconnection between CDN-a and CDN-b for the considered request. CDN-b is a downstream CDN of CDN-a. And there is one CDNI DRIS interface between CDN-a and CDN-b, bootstrapped with configuration information exchanged over the CDNI control interface during the initial CDN interconnection activation process (see [I-D.ietf-cdni-framework]).
- o In the same way there is one CDNI agreement, one CDN interconnection and one CDNI DRIS interface between CDN-b and CDN-c. CDN-c is a downstream CDN of CDN-b for the considered request.
- o CDN-a receives a content request and decides CDN-b will handle it. CDN-b decides the content request will be handled by CDN-c. CDN-c delivers the content to the user agent from one of its delivery nodes (cache hit).

4.2.1. Case of iterative CDNI request redirection

Figure 3 presents the case where the iterative CDNI request redirection mode is applied between CDN-a and CDN-b, and between CDN-b and CDN-c.

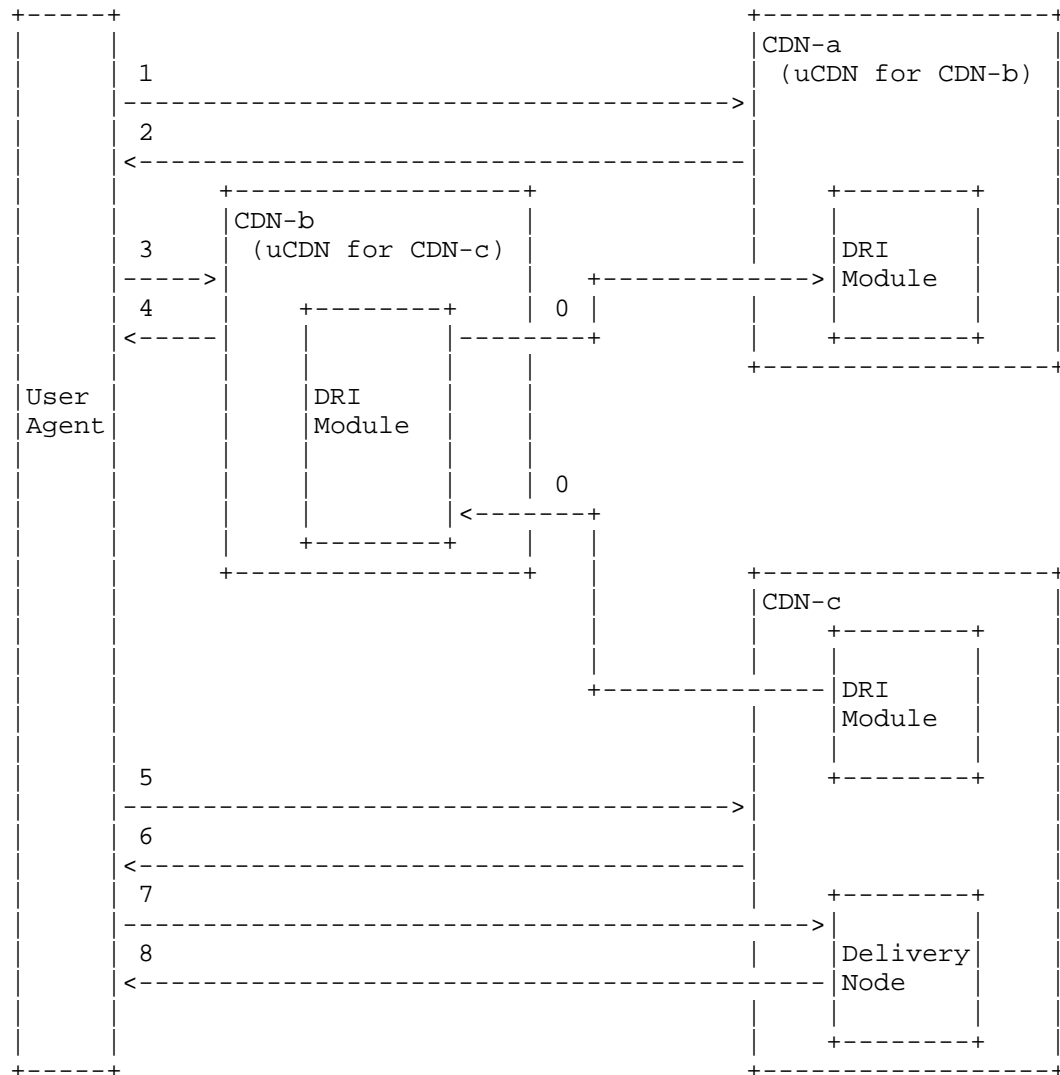


Figure 3: Iterative CDNI request redirection

The operations shown in Figure 3 are as follows:

0. CDN-a gets from CDN-b, over the CDNI DRIS interface between CDN-a and CDN-b, a DRI valid for any content request within their mutual CDNI agreement, and possibly before any user agent content request is received by CDN-a.

In the iterative case, the CDNI DRIS interface may indeed provide the upstream CDN with a unique DRI valid in the frame of their mutual CDNI agreement for any content request received by the upstream CDN.

Basically, it means that the downstream CDN informs the upstream CDN that, when the upstream CDN decides to redirect a content request to this downstream CDN in the frame of their mutual CDNI agreement, the upstream CDN just has to use systematically that DRI to forge the response sent by the upstream CDN to redirect the content request to that downstream CDN.

For example, in case the CDNI request redirection mechanism is based on HTTP 302 redirect messages, this DRI includes a distinguished CDN-domain, which is to be "stacked" in front of the original URL to construct the new URL to redirect the content request to the request routing system of the downstream CDN, as detailed in [I-D.ietf-cdni-framework].

This static and unique DRI for a CDNI agreement set between CDN-a and CDN-b can be exchanged even before the first content request received by CDN-a. This is an example of asynchronous operation over the CDNI DRIS interface. And this is one case where it may be relevant to implement the CDNI DRIS interface "manually"; for example the DRI can be exchanged on the phone or by email simply and only once for the whole duration of the CDNI agreement between CDN-a and CDN-b.

We describe the other operations of Figure 3 below.

0. The same way, CDN-b gets from CDN-c, over the CDNI DRIS interface between CDN-b and CDN-c, a DRI valid for any content request within their mutual CDNI agreement, and possibly before any user agent content request is received by CDN-b.

1. The considered content request, sent by a user agent (and/or by an intermediate local DNS in case it is a DNS request), arrives at CDN-a.

2. CDN-a decides this content request is to be best handled by CDN-b. Therefore, CDN-a sends a response to the content request, forged with the DRI provided by CDN-b.

3. A request is thus sent by the user agent (and/or by an intermediate local DNS in case it is a DNS request) to CDN-b.
4. CDN-b decides this content request is to be best handled by CDN-c. Thus, CDN-b sends a response to the content request forged with the DRI provided by CDN-c.
5. A request is sent by the user agent (and/or by an intermediate local DNS in case it is a DNS request) to CDN-c.
6. CDN-c decides it will handle the request, i.e. it will be the one to deliver the content to the user agent. CDN-c selects one of its delivery nodes to handle that content request. CDN-c generates a response redirecting the content request to this delivery node.
7. The user agent emits a content request to the selected delivery node.
8. The selected delivery node from CDN-c delivers the requested content to the user agent.

4.2.2. Case of recursive CDNI request redirection

Figure 4 presents the case where the full-recursive CDNI request redirection mode is applied between CDN-a and CDN-b, and between CDN-b and CDN-c.

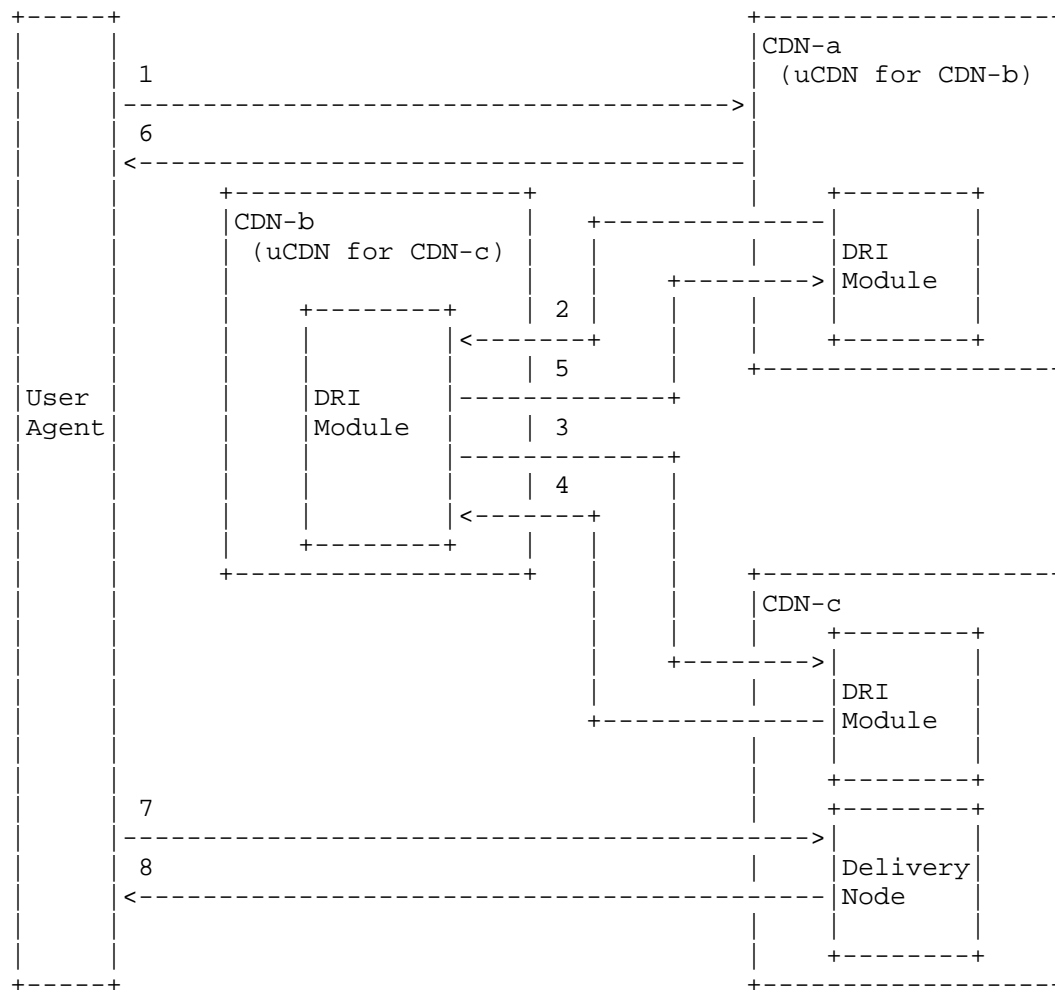


Figure 4: Full-recursive CDNI request redirection

The operations shown in Figure 4 are as follows:

1. The considered content request, sent by a user agent (and/or by an intermediate local DNS in case it is a DNS request), arrives at CDN-a.
2. CDN-a decides this content request is to be best handled by CDN-b. CDN-a sends a request to CDN-b over the CDNI DRIS interface set up between CDN-a and CDN-b. The request includes all information about the content request CDN-b needs in order to select the most

appropriate delivery resource and to response to CDN-a with the corresponding adequate DRI.

3. CDN-b decides this content request is to be best handled by CDN-c. CDN-b sends a request to CDN-c over the CDNI DRIS interface set up between CDN-b and CDN-c. The request includes all information about the content request CDN-c needs in order to select the most appropriate delivery resource and to response to CDN-b with the corresponding adequate DRI.

4. CDN-c decides it will handle the content request, i.e. it will be the one to deliver the content to the user agent. CDN-c selects one of its delivery nodes to handle that content request. CDN-c sends a response to CDN-b, over the CDNI DRIS interface between CDN-b and CDN-c, with a DRI corresponding to that delivery node.

5. CDN-b sends a response to CDN-a, over the CDNI DRIS interface between CDN-a and CDN-b, with a DRI corresponding to the selected delivery node from CDN-c. This DRI is based on the DRI provided by CDN-c. Details about the manipulations achieved by CDN-b over the DRI received from CDN-c to generate the DRI sent to CDN-a are to be provided in the dedicated specification of the CDNI DRIS interface.

6. CDN-a sends a response to the content request, forged with the DRI provided by CDN-b.

7. The user agent emits a content request directly to the selected delivery node from CDN-c.

8. The selected delivery node from CDN-c delivers the requested content to the user agent.

The Figure 5 below presents the case where the semi-recursive CDNI request redirection mode is applied between CDN-a and CDN-b, and between CDN-b and CDN-c.

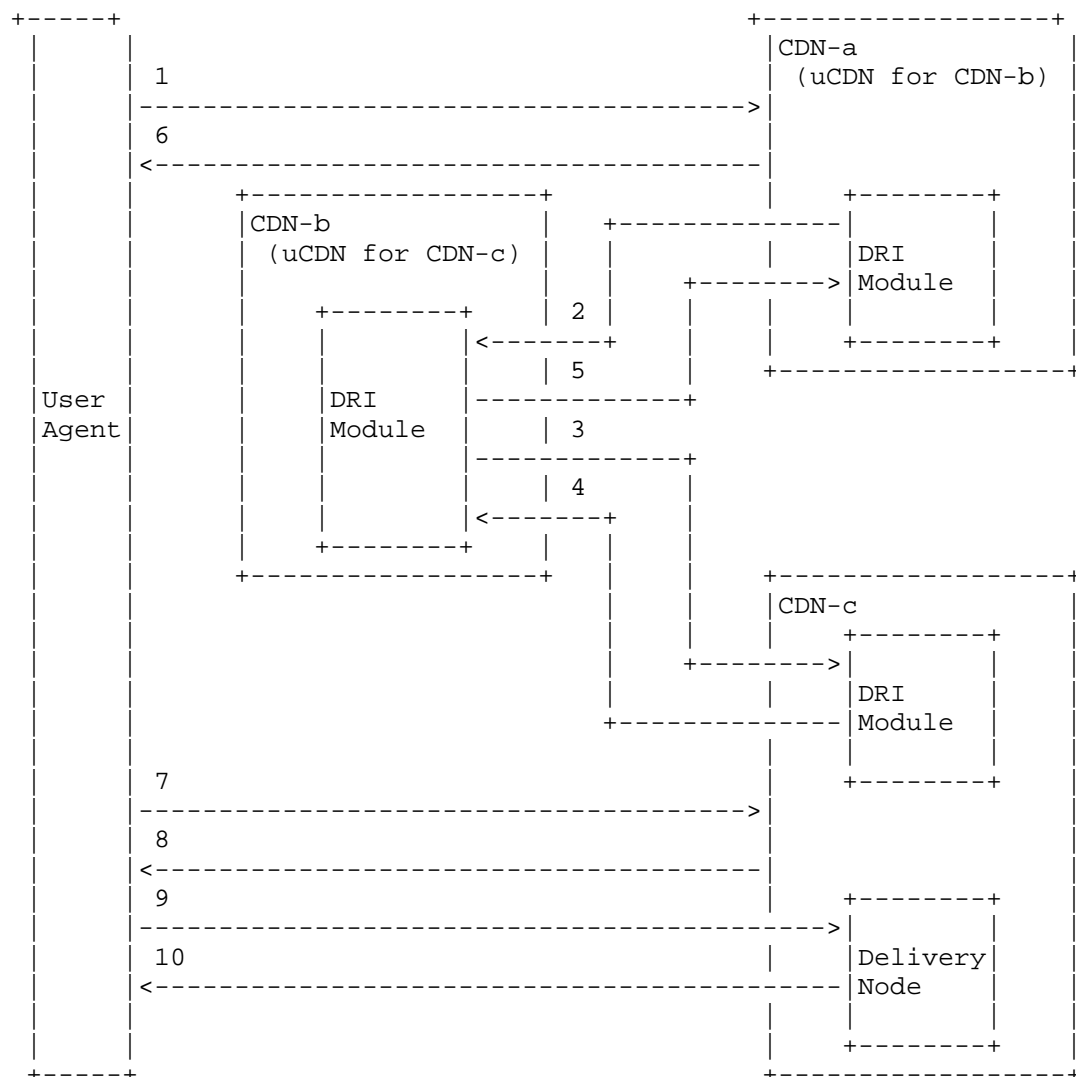


Figure 5: Semi-recursive CDNI request redirection

The operations shown in Figure 5 are as follows:

1. The considered content request, sent by a user agent (and/or by an intermediate local DNS in case it is a DNS request), arrives at CDN-a.
2. CDN-a decides this content request is to be best handled by

CDN-b. CDN-a sends a request to CDN-b over the CDNI DRIS interface set up between CDN-a and CDN-b. The request includes all information about the content request CDN-b needs in order to select the most appropriate delivery resource and to response to CDN-a with the corresponding adequate DRI.

3. CDN-b decides this content request is to be best handled by CDN-c. CDN-b sends a request to CDN-c over the CDNI DRIS interface set up between CDN-b and CDN-c. The request includes all information about the content request CDN-c needs in order to select the most appropriate delivery resource and to response to CDN-b with the corresponding adequate DRI.

4. CDN-c decides it will handle the content request, i.e. it will be the one to deliver the content to the user agent. CDN-c sends a response to CDN-b, over the CDNI DRIS interface between CDN-b and CDN-c, with a DRI corresponding to its request routing system.

5. CDN-b sends a response to CDN-a, over the CDNI DRIS interface between CDN-a and CDN-b, with a DRI corresponding to the selected delivery resource (i.e. to the request routing system of CDN-c). This DRI is based on the DRI provided by CDN-c. Details about the manipulations achieved by CDN-b over the DRI received from CDN-c to generate the DRI sent to CDN-a are to be provided in the dedicated specification of the CDNI DRIS interface.

6. CDN-a sends a response to the content request, forged with the DRI provided by CDN-b.

7. The content request is redirected by the user agent (and/or by an intermediate local DNS in case it is a DNS request) to the selected delivery resource, i.e. to CDN-c.

8. CDN-c decides it will handle the content request, i.e. it will be the one to deliver the content to the user agent. CDN-c selects one of its delivery nodes to handle that content request. CDN-c generates a response redirecting the content request to this delivery node.

9. The user agent emits a content request to the selected delivery node.

10. The selected delivery node from CDN-c delivers the requested content to the user agent.

As shown on Figure 4 and 5, when the recursive mode is applied, the upstream CDN may have to send a request over a CDNI DRIS interface every time it receives a content request. This is an example of

synchronous operations over the CDNI DRIS interface. Having one CDNI DRIS request per content request may lead to scalability issues, even if the CDNI DRIS interface is implemented with a network protocol rather than "manually". Yet these issues can be solved. This is to be documented in the dedicated specification of the CDNI DRIS interface.

4.3. Key points to note about the CDNI DRIS interface

To conclude this section, the key points to note about the CDNI DRIS interface are the following:

- o The CDNI DRIS interface MUST always exist between two interconnected CDNs. Whatever the CDNI request redirection strategy applied (recursive or iterative), the CDNI DRIS interface exists, even if it is implemented "manually" in some cases, as suggested in the example of Figure 3 above.
- o Moreover the CDNI DRIS interface MUST be unique from a functional perspective. Another way to say this is that, from a functional perspective, this is the exact same CDNI DRIS interface that is used, and the exact same type of information that are exchanged over this interface, whatever the CDNI request redirection strategy applied (recursive or iterative).
- o The specification of the CDNI DRIS interface MUST enable synchronous and asynchronous CDNI DRIS operations: operations possibly triggered by the CDN selection process for a given content request (as shown on Figure 4 and 5), as well as operations achieved independently of any content request (as shown on Figure 3).
- o The specification of the CDNI DRIS interface MUST enable operations initiated by uCDN (as shown on Figure 4 and 5) or by dCDN (as shown on Figure 3).
- o The specification of the CDNI DRIS interface MUST be compatible with "automatic" (i.e. "with a specific network protocol") and "manual" implementations, as discussed in Section 4.2. The specification of the CDNI DRIS interface SHALL include a description for each of these implementation options.
- o The CDNI routing and CDNI DRIS interfaces MUST be independent.

5. CDNI request routing is just another routing and signaling problem

This part of the CDN interconnection framework the IETF has been referring to so far with the term "CDNI Request Routing" is just another routing, signaling and forwarding problem in a long series of the telecommunication history.

For example one can draw an analogy with the IP/MPLS-TE framework, notably, as shown on Figure 6 and below, between:

- o the CDNI routing interface and any IP routing protocol/interface (such as BGP/IGP IP routing protocols),
- o the CDNI DRIS interface and a signaling protocol such as RSVP-TE.

It is to be noted that such an analogy must be considered with care because the context and objectives of CDN interconnection are different from IP routing, signaling and forwarding. In particular, the type of information exchanged via the CDNI DRIS interface is totally different from the information exchanged by protocols like RSVP-TE in IP/MPLS networks. Anyway this analogy may ease to understand the role of the CDNI DRIS interface in the CDNI framework.

Another key point to note in this analogy is that IP prefix is one of the most important concepts in IP networks. It is used in most IP processes (IP routing, packet filtering, MPLS-TE signaling, etc.). It has been a highly useful and powerful concept to simplify the specification of TCP/IP protocols as well as to ensure performance and scalability of IP networks. For example exchanging IP routing information per IP prefix has been a MUST to ensure IP routing performance and scalability.

The equivalent concept to IP prefix has the same importance in the CDN interconnection framework. The concept is named "contentRequestScope" in Figure 6. A contentRequestScope designates a class of content requests sharing common properties (e.g., "all the HTTP requests", or "all HTTP requests issued by French users", or "all RTSP and HTTP requests with signed URL", etc.).

In the same way as IP prefix is a key concept for specifying TCP/IP protocols, "contentRequestScope" is THE main concept for the CDN interconnection framework. This highly useful and powerful concept SHALL be used to simplify the specification of ALL CDN interconnection interfaces, as well as to ensure performance and scalability in CDN interconnection.

IP/MPLS-TE concepts	CDN Interconnection concepts
IP packet Forwarding	Content request redirection
IP prefix (correspond to a class of IP addresses sharing the same prefix)	ContentRequestScope (correspond to a class of content requests sharing common properties)
Ingress Provider Edge Router (ILSR)	upstream CDN (uCDN)
Egress Provider Edge Router (ELSR)	Delivery resource (delivery node or downstream CDN)
IP routing interfaces (running IGP/BGP IP routing protocols)	CDNI routing interface
IP routing information (IP prefix <=> Next Hop)	CDNI routing information (ContentRequestScope <=> dCDN)
IP FEC table construction (IP prefix <=> ELSR)	Redirection table construction (ContentRequestScope <=> delivery resource)
Signaling information provided by the RSVP-TE protocol (MPLS label <=> ELSR)	Signaling information provided by the CDNI DRIS interface (DRI <=> delivery resource)
To encapsulate an IP packet in an MPLS frame (using an MPLS Label)	To generate a redirected content request (using a DRI)
To forward a packet out of the nominal route based on IGP and BGP routing information, thanks to MPLS Traffic Engineering technologies	To redirect a content request out of the authoritative CDN, to another CDN, thanks to CDN interconnection technologies
To forward an IP packet encapsulated in an MPLS frame via - or to - the destination ELSR thanks to the MPLS header of the MPLS frame	To redirect a content request to to the selected downstream CDN - or to the selected delivery node - thanks to the Downstream Resource Identifier (DRI) used to forge the response to the initial content request

Figure 6: Analogy between IP/MPLS-TE and CDNI concepts

CDNI Routing interfaces exchanges CDNI routing information per contentRequestScope, i.e. per class of content requests sharing common properties.

Analogically: IP routing protocols like IGP/BGP exchange routing information per IP prefix, i.e. per class of IP addresses sharing common properties.

The upstream CDN builds its redirection table with information provided by the CDNI Routing interface and the CDNI DRIS interface.

Analogically: The Ingress Provider Edge Router (ILSR) builds its forwarding table with information provided by the IP routing and MPLS-TE signaling protocols.

When receiving a content request, which has possibly been already redirected by some upstream CDNs, a CDN consults its forwarding table to generate a response using the adequate DRI, provided by the CDNI DRIS interface, which redirects the content request to the selected delivery resource (which is a CDN or a node downstream).

Analogically: When receiving an IP packet, possibly encapsulated in an MPLS frame, an Ingress Provider Edge Router (ILSR) consults its forwarding table to manipulate it adequately. This manipulation may include to encapsulate it in an MPLS frame, based on the signaling information provided by the RSVP-TE protocol, which makes it being then forwarded to the right Egress Provider Edge router (ELSR).

6. Conclusion and recommendations

A first key message of this document is that the term "CDNI request routing" is confusing as it makes a mix-up of clearly different set of processes and interfaces.

A second key message of this document is that this part of the CDN interconnection framework the IETF has been referring to so far with the term "CDNI Request Routing" is just another routing, signaling and forwarding problem in a long series of the telecommunication history. For example, one can draw a direct analogy between the IP/MPLS-TE framework and the CDN interconnection framework.

Thus one can directly be inspired by the best practices and results of the efforts invested over years, even decades, to develop technologies in the past, such as IP/MPLS-TE, to specify adequately CDN interconnection interfaces.

The proposals from the current document can be smoothly integrated in the WG drafts, especially [I-D.ietf-cdni-framework] and [I-D.ietf-cdni-requirements], as they essentially propose (useful) clarifications of the existing framework.

We recommend to review [I-D.ietf-cdni-framework], at least its section 4.2. Appendix A provides a proposal to replace the current Section 4.2 of [I-D.ietf-cdni-framework].

We recommend to review [I-D.ietf-cdni-requirements], to clearly distinguish the requirements specific to the CDNI routing interface from the requirements specific to the CDNI DRIS interface.

We also recommend to review the charter of the CDNI WG, in particular to replace the following objective:

- o "WG Creation + 18 months: Submit specification of the CDNI Request Routing protocol to IESG as Proposed Standard"

by the two following objectives:

- o "WG Creation + 18 months: Submit specification of the CDNI Routing interface to IESG as Proposed Standard"
- o "WG Creation + 18 months: Submit specification of the CDNI Downstream Resource Identifier Signaling (DRIS) Interface to IESG as Proposed Standard"

Last but not least, we recommend that the specification of ALL CDN interconnection interfaces, including the CDNI routing interface and the CDNI DRIS interface, rely on the "contentRequestScope" concept, i.e. the equivalent concept to IP prefix for CDN interconnection. In the same way as IP prefix is a key concept for specifying TCP/IP protocols, "contentRequestScope" is THE main concept for the CDN interconnection framework. This highly useful and powerful concept SHALL be used to simplify the specification of ALL CDN interconnection interfaces, as well as to ensure performance and scalability in CDN interconnection.

7. Acknowledgments

The authors would like to thank Chris Hawinkel, Larry Peterson, Yvan Massot, Ali Gouta, Emile Stephan, and Patrick Fleming for their inputs and comments.

They also thank the contributors of the EU FP7 OCEAN project in the frame of which these proposals have been elaborated.

The work leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) in OCEAN project (FP7-ICT-248775; <http://www.ict-ocean.eu/>).

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

Those are discussed in [I-D.ietf-cdni-problem-statement].

10. Informative References

[I-D.ietf-cdni-framework]

Peterson, L. and B. Davie, "Framework for CDN Interconnection", draft-ietf-cdni-framework-00 (work in progress), April 2012.

[I-D.ietf-cdni-problem-statement]

Niven-Jenkins, B., Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", draft-ietf-cdni-problem-statement-08 (work in progress), June 2012.

[I-D.ietf-cdni-requirements]

Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", draft-ietf-cdni-requirements-03 (work in progress), June 2012.

[I-D.ietf-cdni-use-cases]

Bertrand, G., Emile, S., Burbridge, T., Eardley, P., Ma, K., and G. Watson, "Use Cases for Content Delivery Network Interconnection", draft-ietf-cdni-use-cases-08 (work in progress), June 2012.

Appendix A. Proposal for Section 4.2 of CDNI framework draft

"4.2. Request Routing Interface

The request routing interface is not a single interface, it corresponds to two separate interfaces with clear roles:

1) The CDNI Routing Interface

The CDNI routing interface is dedicated to the CDNI routing process. The CDNI routing process consists in the advertisement of CDNI routing information from the downstream CDN to the upstream CDN (the upstream CDN never provides the downstream CDN with any CDNI routing information). CDNI routing information details capabilities of the downstream CDN that the upstream CDN must take into account in its dCDN selection process.

2) The CDNI Downstream resource Identifier Signaling (DRIS) Interface

AFTER the CDN selection process is achieved by the upstream CDN for a given content request, the upstream CDN MUST generate a response redirecting that request towards the selected delivery resource by inserting in this response an identifier of that selected delivery resource.

This selected delivery resource can be:

1. a delivery node within the upstream CDN,
2. a downstream CDN (i.e. the request routing system of that downstream CDN), or
3. a delivery node within a downstream CDN.

In the cases "2." and "3." , the identifier of that selected delivery resource MUST be provided by the downstream CDN to the upstream CDN. The downstream CDN uses the CDNI DRIS interface to provide the upstream CDN with this identifier called "Downstream resource Identifier" (DRI).

Authors' Addresses

Yannick Le Louedec
France Telecom - Orange
2 avenue Pierre Marzin
Lannion, 22307
France

Phone: +33 2 96 05 17 64
Email: yannick.lelouedec@orange.com

Anne Marrec
France Telecom - Orange
2 avenue Pierre Marzin
Lannion, 22307
France

Phone: +33 2 96 05 18 71
Email: anne.marrec@orange.com

Gilles Bertrand
France Telecom - Orange
38-40 rue du General Leclerc
Issy les Moulineaux, 92130
France

Phone: +33 1 45 29 89 46
Email: gilles.bertrand@orange.com

Marcin Pilarski
Orange Polska / WUT
ul. Obrzezna 7 / ul. Koszykowa 75
Warsaw, Mazowieckie 02-691 / 00-662
Poland

Email: marcin.pilarski@orange.com / marcin.pilarski@mini.pw.edu.pl

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 17, 2013

K. Ma
Azuki Systems, Inc.
July 16, 2012

Content Distribution Network Interconnection (CDNI) Metadata Interface
draft-ma-cdni-metadata-03

Abstract

Content publishers (CPs) often use multiple Content Delivery Networks (CDNs) to deliver content to consumers. Though existing interactions between CPs and individual CDNs are beyond the scope of CDN interconnection (CDNI), it is important to understand the management capabilities and features available with existing non-interconnected multi-CDN deployments. Before migrating to CDNI, CPs must first assess the suitability of CDNI as a replacement for their existing non-interconnected multi-CDN deployments. CDN feature configuration and capability advertisement and enforcement is likely to occur through the CDNI metadata interface (MI). This document describes an approach to implementing the CDNI MI through the use of an extensible metadata model and a light-weight HTTP-based API.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 17, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	4
1.1.	Terminology	5
1.2.	Abbreviations	5
2.	CDNI Metadata Data Model	6
2.1.	Domain Table	7
2.2.	Base Address Table	7
2.2.1.	Hierarchical Base Addresses	9
2.3.	Agent Table	9
2.4.	Metadata Table	11
2.4.1.	Hierarchical Metadata	12
3.	CDNI Metadata Bootstrapping	14
4.	CDNI Metadata Management	15
4.1.	Metadata API	16
4.1.1.	Metadata Creation	17
4.1.2.	Metadata Update	18
4.1.3.	Metadata Refresh Trigger	19
4.1.4.	Metadata Retrieval	20
4.1.5.	Metadata Removal	22
4.1.6.	Metadata Errors	23
4.1.7.	Metadata Prepositioning	23
5.	Metadata Definitions	24
5.1.	Origin Server	24
5.2.	Activation Time	24
5.3.	Deactivation Time	25
5.4.	Administrative Disable	25
5.5.	Delegation Depth	26
5.6.	Footprint Filter	26
5.7.	HTTP Header Filter	27
5.8.	HTTP Header Logging	27
5.9.	Protocol Filter	27
5.10.	SSL Required	28
5.11.	SSL Client Authentication Required	28

5.12. URL Hash	29
6. IANA Considerations	29
7. Security Considerations	29
8. Acknowledgements	30
9. Appendix A: Domain API	30
9.1. Domain Creation	31
9.2. Domain Update	31
9.3. Domain Retrieval	31
9.4. Domain Removal	32
9.5. Domain Errors	32
10. Appendix B: Agent API	32
10.1. Agent Creation	33
10.2. Agent Update	34
10.3. Agent Retrieval	35
10.4. Agent Removal	35
10.5. Agent Errors	35
11. References	36
11.1. Normative References	36
11.2. Informative References	36
Author's Address	36

1. Introduction

The use cases described in the CDNI use case document [I-D.ietf-cdni-use-cases] provide motivational use cases for CDN interconnection (CDNI). They describe reasons and situations where CDNI provides a benefit to CDN vendors as well as content service providers (CSPs). Additional use cases exist which describe how CDNs are used today, however, these use cases often involve specific features (e.g., customized content transformations, content security, client authentication and filtering, content acquisition optimization and redundancy, etc.) which are beyond the scope of CDNI. Though the features themselves are not relevant to CDNI, the ability to support those features or enforce policies related to those features in a generic and extensible manner should be considered when designing CDNI interfaces. The ability to support feature parity with existing deployment models (i.e., non-CDNI-based CDN federation) may help to remove barriers to CDNI adoption.

Though certain interfaces are out of scope of CDNI, e.g.:

- o upstream CDN (uCDN) configuration by the CP
- o uCDN content acquisition
- o uCDN content delivery
- o downstream CDN (dCDN) content acquisition
- o end user (EU) content acquisition
- o third party workflow management
- o third party request routing

An awareness of these interfaces and an understanding of the restrictions which they may impose on CDNI request routing is useful for understanding the needs of the CDNI metadata interface (MI). As described in the "Dynamic CDNI Metadata Acquisition Example" section in the CDNI framework document [I-D.davie-cdni-framework], upon receiving a request routing interface (RRI) request, the MI MAY be used to retrieve metadata that is "considered" before responding to the RRI request. To that end, the MI MUST define a deterministic method for handling metadata processing. Though the definition and interpretation of any individual piece of metadata is beyond the scope of CDNI, a well-defined method for how to respond to a RRI request when any unknown metadata value is encountered MUST be supported.

This document describes a simple data model for representing CDNI metadata and a simple protocol for creating and retrieving CDNI metadata in an opaque manner. The term opaque, in this case, should be understood to mean: without understanding the underlying meaning or interpretation of the metadata being represented. The metadata model and retrieval protocol SHOULD be completely independent of the definition of individual metadata values. The metadata model and retrieval protocol MUST also define default behaviors for dealing with metadata processing errors. The document defines a list of metadata which are likely applicable to a broad range of CDNI deployments. The document also provides a separate list of metadata which are likely to be desirable to content publishers (CPs). This document is not intended to suggest that any additional interfaces or requirements are needed beyond those already specified in the CDNI requirements document [I-D.ietf-cdni-requirements], nor is this document intended to suggest that any out of scope interfaces or content publisher feature functionality should be brought into scope. The metadata examples provided are intended only to illustrate possible features that interconnected CDNs may wish to support and the extensibility of the metadata model to handle those situations.

1.1. Terminology

[Ed. insert terminology reference]

1.2. Abbreviations

- o CDN: Content Distribution Network
- o uCDN: Upstream Content Distribution Network
- o dCDN: Downstream Content Distribution Network
- o CDNI: Content Distribution Network Interconnection
- o CP: Content Publisher
- o CSP: Content Service Provider
- o EU: End User
- o NSP: Network Service Provider
- o RRI: Request Routing Interface
- o MI: Metadata Interface

- o CI: Control Interface

2. CDNI Metadata Data Model

The simple data model is shown in Figure 1 below. It includes a top level Domain object which describes the site(s) to which metadata is associated. The term site, in this case, should be understood to mean a collection of related content assets accessed through a single portal or Web-site. The Domain is associated with zero or more opaque Metadata objects. Each Metadata object is associated with one or more Base Address objects. The Metadata objects are each associated with a URI extension, applicable to any of the associated Base Addresses. A combination of Base Address and URI prefix matching is used identify Metadata to allow for hierarchical associations between individual Metadata and sets of content items. Each Domain is also associated with one or more Agent objects. Agents represent entities which require access to metadata (e.g., CPs, uCDNs, dCDNs, or local operators). An Agent is associated with each Metadata entry allowing different Metadata values to be returned to different Agents.

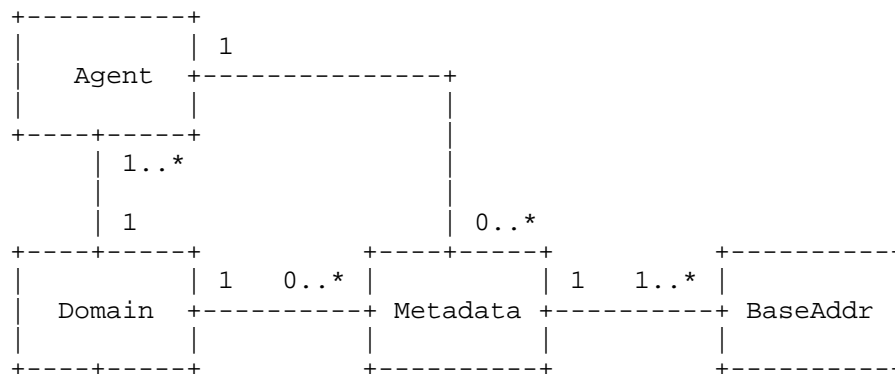


Figure 1: CDNI Metadata Data Model

Note: The data model described above provides the basic components required for distributing Metadata and implementing the CDNI MI. The specific semantics of individual pieces of metadata are abstracted to allow for opaque distribution of metadata. Not all of the information described need be distributed through the MI. Some information (e.g., Domains and Agents) may be necessary for the MI to function, but MAY be negotiated or implemented out-of-band. They could be configured either by the CDN as part of a non-CDNI process, or through the CDNI control interface (CI) bootstrapping process, or using the MI APIs described herein. The MI APIs may also be used by

CDNs, internally, to configure themselves. The complete data model and full set of APIs are provided as part of a holistic MI description.

The following sections describe an example implementation of the metadata scheme described above using a standard SQL database.

2.1. Domain Table

The Domain object contains basic information related to the site being described. The example shown contains a primary key index and a unique name for the site. An OPTIONAL site description (e.g., a textual description of the site and its content) and site provider (e.g., the name of the CP or CSP which owns the content) information is also included.

```
CREATE TABLE "domain" ("domain_id" serial primary key,  
                        "name" character varying(255) NOT NULL,  
                        "provider" character varying(255),  
                        "description" character varying(4095));  
CREATE UNIQUE INDEX index_domain ON domain (name);
```

The Domain is the central object for binding Metadata. The example Domain shown below highlights the descriptive nature of the Domain object:

```
domain_id: 1  
name: acme  
provider: acme rocket-powered products, inc  
description: fine purveyors of high quality anvils, rubber bands,  
            bird seed, and rocket-powered footwear.
```

2.2. Base Address Table

The Base Address object contains basic hostname and base URI information related to the site being accessed. The example shown requires a primary key index, a string containing the hostname and base URI, and a foreign key reference to the Metadata to which this Base Address is associated. A uniqueness constraint is imposed on baseaddr/metadata_id pairs to prevent duplicate Base Address entries for a given Metadata.

```
CREATE TABLE "baseaddr" ("baseaddr_id" serial primary key,  
                          "baseaddr" character varying(255) NOT NULL,  
                          "metadata_id" integer NOT NULL);  
CREATE UNIQUE INDEX index_baseaddr ON baseaddr (baseaddr,  
                                                metadata_id);
```

Base Address Table Definition

The Base Address objects allows multiple hostname and base URI pairs to be associated with each Metadata object denoting the list of Base Addresses through which content within the Domain may be accessed. There are many cases where different Base Addresses are used to access the same content, e.g.:

- o internal vs. external addresses: content may be accessible via both internal 10-net IP addresses and their associated DNS addresses and base URIs, as well as publicly routable external IP addresses and their associated DNS addresses and base URIs, where all of the addresses point to the same content servers and the base URIs are mapped to the same base directories,
- o service white-labeling: multiple CSPs may provide access to the same content through different branded services where each branded service has its own DNS address and/or base URI, but all of the services point to the same content, or
- o analytics partitioning: redirects from other sites may use different DNS addresses and/or base URIs, so that they may be easily accounted for, while still pointing at the same content.

The example Base Addresses shown below represent two DNS addresses through which content may be accessed as well as an internal IP address which may be used for staging:

```
baseaddr_id: 1
baseaddr: wile.e.coyote.acme.com
metadata_id: 1
```

```
baseaddr_id: 2
baseaddr: road.runner.acme.com
metadata_id: 1
```

```
baseaddr_id: 3
baseaddr: 10.10.10.10/meemeep
metadata_id: 1
```

Note: The exact schema described above may result in heavy duplication of Base Addresses. It is presented as an example for its simplicity, however, it may be optimized by using other table joining implementation schemes.

2.2.1. Hierarchical Base Addresses

In order to support hierarchical Base Addresses, the wildcard '*' SHOULD be allowed as the first part of DNS-type Base Addresses. The wildcard does not make sense at the beginning of IP Address-type Base Addresses. Though a wildcard at the end of IP Address-type Base Addresses would make more sense, support for IP Address-type Base Addresses is OPTIONAL. The wildcard signifies the applicability of the associated Metadata value to all Base Addresses which match the address suffix.

The following two Base Addresses condense the previous example by allowing all acme.com DNS addresses:

```
baseaddr_id: 1
baseaddr: *.acme.com
metadata_id: 1

baseaddr_id: 2
baseaddr: 10.10.10.10/meemeep
metadata_id: 1
```

Note: There is no explicit enforcement that Base Addresses associated with a given piece of Metadata not overlap, however, for performance reasons, Base Addresses associated with a given piece of Metadata SHOULD NOT be allowed to overlap.

2.3. Agent Table

The Agent object contains basic information for authenticating entities which require access to Metadata. The example shown contains a primary key index, a string containing the username, an OPTIONAL string containing the password (possibly hashed or encrypted), a boolean value for differentiating between full read/write access (e.g., for uCDNs) and read only access (e.g., for dCDNs), and a foreign key reference to the Domain to which this Agent is associated. A uniqueness constraint is imposed on username/domain_id pairs to prevent duplicate Agent entries for a given Domain.

```
CREATE TABLE "agent" ("agent_id" serial primary key,
                      "username" character varying(255) NOT NULL,
                      "password" character varying(255),
                      "read_only" boolean DEFAULT false NOT NULL,
                      "domain_id" integer NOT NULL);
CREATE UNIQUE INDEX index_agent ON agent (username, domain_id);
```

Agent Table Definition

Note: The password field is included to support the HTTP authentication described in the API sections, however, if alternate authentication schemes are used, the password may not be necessary.

The Agent objects manage Metadata access rights. The Agent functionality as described attempts to address two issues:

- o security concerns: where unauthorized injection or deletion of Metadata may alter the functionality of a content service and MUST be prevented, as described in the Security Considerations section, and
- o customization requirements: where retrieval of certain metadata may require different responses depending on the Agent who is accessing the Metadata (e.g., with multiple and/or cascaded dCDNs).

Note: Though both of the above issues could be addressed through means outside of the MI, or through a means common to all of the CDNI interfaces, the Agent serves the purpose of addressing these needs within the context of the MI, in lieu of a consensus alternative and as per the CDNI framework document [I-D.davie-cdni-framework].

The example Agents shown below represent a uCDN Agent with write privileges and two dCDN Agents with read-only permissions:

```
agent_id: 1
username: ucdn
password: xxx
read_only: false
domain_id: 1
```

```
agent_id: 2
username: dcdn1
password: yyy
read_only: true
domain_id: 1
```

```
agent_id: 3
username: dcdn2
password: zzz
read_only: true
domain_id: 1
```

2.4. Metadata Table

The Metadata object contains the actual individual pieces of metadata for the site being described. The example shown contains a primary key index, a string containing the URI(s) to which the metadata applies, a name/value pair of strings which represent the name and value of the Metadata, respectively, as well as a boolean value stating whether or not the given Metadata must be enforced. An OPTIONAL priority value is included for creating order lists of values for a given named Metadata. An OPTIONAL ttl value and timeout field are included to support metadata invalidation. The table also contains a foreign key reference to the Domain to which this Metadata is associated and a foreign key reference to the Agent to whom this Metadata is intended. A compound uniqueness constraint is also applied to each uri/name/priority/domain_id/agent_id tuple to prevent a given Metadata from being ambiguously applied multiple times to the same URI in a given Domain for a given Agent.

```
CREATE TABLE "metadata" ("metadata_id" serial primary key,
                           "uri" character varying(4095) NOT NULL,
                           "name" character varying(511) NOT NULL,
                           "value" character varying(65535) NOT NULL,
                           "must_enforce" boolean DEFAULT true NOT NULL,
                           "priority" integer DEFAULT 0 NOT NULL,
                           "ttl" integer DEFAULT 0 NOT NULL,
                           "timeout" timestamp without time zone,
                           "domain_id" integer NOT NULL,
                           "agent_id" integer NOT NULL);
CREATE UNIQUE INDEX index_metadata ON metadata (uri, name, priority,
                                                domain_id, agent_id);
```

The name/value pair is represented as simple opaque strings. The MI does not require an understanding of the semantics or inherent meaning of Metadata names or values to distribute the Metadata. Though, each piece of Metadata MUST have a defined set of semantics in order to be enforced, distributing the Metadata and determining whether or not the Metadata is supported does not require any understanding of the Metadata semantics, but rather, only an ability to identify supported Metadata by their name is REQUIRED. Metadata names SHOULD be properly defined and registered, and any implied functionality SHOULD be agreed upon and documented. A base set of CDNI Metadata is provided in the Metadata Definitions Section.

The intent of the must_enforce boolean is to identify Metadata that MUST be enforced by all CDNs. If a CDN is unable to understand or is unable to comply with the Metadata, it MUST NOT deliver the content being requested. For dCDNs, the must_enforce flag defines how to respond to MI and RRI requests when unknown or unsupported Metadata

is encountered. If Metadata is marked as `must_enforce`, then the dCDN MUST NOT accept any RRI request if it is unable to enforce that piece of Metadata (e.g., the named Metadata is not supported, the Metadata value is invalid, or the Metadata value is not supported). If the MI request resulted from a "recursive" RRI request, then the dCDN MUST return an error to the uCDN. If the MI request resulted from an "iterative" RRI request, then the dCDN MUST respond with a 403 Forbidden status code to the EU and report the failure to the uCDN.

In the case of cascaded CDN deployments, though a given CDN may not be able to enforce a given piece of Metadata, other CDNs further down stream may be able to enforce that Metadata. When a Metadata rejection occurs, the CDN SHOULD still store the Metadata so that it can be provided to other dCDNs.

The OPTIONAL priority value is provided to allow configuration of ordered Metadata lists. When specifying multiple values for a given named Metadata, each value MUST be specified with a unique priority value. The explicit priority value enforces a deterministic ordering across MI implementations.

The OPTIONAL ttl value is provided to allow configuration of a Metadata TTLs. If the ttl is specified, it MUST be specified in seconds and the timeout field SHOULD be populated by the local MI processor and used internally, to prevent the need for clock synchronization between MI processors.

The association of each Metadata to an Agent allows different Agents to retrieve different Metadata values for a given URI in the given Domain. This is intended to allow CDNs to separate upstream Metadata from downstream Metadata (e.g., a uCDN content acquisition URL may point to a CP origin, however, the content acquisition URL that the dCDN retrieves from the uCDN may point at a surrogate in the uCDN; likewise the content acquisition URLs for different dCDNs may point at different surrogates in the uCDN). Though this information could be hidden within a CDN's implementation, the security aspects related to deterministically associating an authenticated Agent with the proper metadata should be considered as part of the MI. Explicitly representing this in the data model reduces ambiguity in Metadata retrieval.

2.4.1. Hierarchical Metadata

In order to support hierarchical metadata, `/'*` SHOULD be allowed as the last part of the URI hierarchy, signifying the application of this Metadata value to all URIs which match this URI prefix. If multiple Metadata are defined with overlapping prefixes, the URI with the longest prefix match MUST be used. The uniqueness constraint on

the uri/name/priority/domain_id tuple allows for unambiguous resolution of Metadata priority.

Note: The wildcard is only supported at the end of the URI string to provide a well-defined ordering for URI prefixes (i.e., longest prefix matching). Use of generalized regular expression matching requires ordering rules to ensure deterministically coherent results across multiple MI implementations. It is assumed that the URI path extensions (beyond the base paths provided in the Base Address) for content will be the same across CDNs. Any CDN specific URL rewrites MUST only affect the Base Address portion of the URL as defined in the Base Address.

Note: It is often desirable to separate specific types of files which may live in the same directory (e.g., .m3u8 vs .ts). Wildcard support in the URI support for file extension differentiation, i.e., `'/*[.extension]'`, is OPTIONAL.

Given the following four Metadata objects, the value of color is defined five times, for three different URIs, all within the same Domain, but for different Agents:

```
metadata_id: 1
uri: /*
name: color
value: blue
must_enforce: false
priority: 0
ttl: 0
domain_id: 1
agent_id: 2
```

```
metadata_id: 2
uri: /*
name: color
value: gold
must_enforce: false
priority: 0
ttl: 0
domain_id: 1
agent_id: 1
```

```
metadata_id: 3
uri: /*
name: color
value: blue
must_enforce: false
priority: 1
```

```
ttl: 0
domain_id: 1
agent_id: 1

metadata_id: 4
uri: /grass/*
name: color
value: brown
must_enforce: false
priority: 0
ttl: 0
domain_id: 1
agent_id: 2

metadata_id: 5
uri: /grass/on/the/other/side/*
name: color
value: green
must_enforce: false
priority: 0
ttl: 0
domain_id: 1
agent_id: 2
```

The default value for the color metadata (signified by the all encompassing URI `"/"`) is blue for the dCDN Agent and gold for the uCDN Agent, though the default color may be blue for the uCDN as well (as signified by the lower priority alternate color value). Alternate colors are associated with requests from the dCDN Agent for URIs that begin with `"/grass"`. By default `"/grass"` has a color of brown, except when requesting `"/grass/on/the/other/side/"` which is green.

3. CDNI Metadata Bootstrapping

It is assumed that a well-known hostname to which MI requests should be sent is configured through the CDNI bootstrap data. Bootstrap information is sent through the CDNI CI, as described in the CDNI requirements document [I-D.ietf-cdni-requirements]. The MI APIs described herein are intended to be serviced by the MI running on that host.

Domain and Agent configurations must exist prior to Metadata creation/retrieval. Domains and Agents MAY be created as a part of an off-line business negotiation process or as a part of the CDNI bootstrapping process. Domain and Agent API descriptions are included in Appendix A and Appendix B, respectively. When the Domain

and Agent APIs described are used, access to the APIs SHOULD be secured using SSL with client authentication as described in the Security Considerations section.

Two sets of Agent configurations are also REQUIRED:

- o Upstream Agent Configuration: Agent credentials for all external agents who require access to the local CDN MI, e.g. for dCDNs to retrieve Metadata or for uCDNs to trigger Metadata.
- o Downstream Agent Configuration: Agent credentials for the local CDN to use when accessing uCDN MIs for retrieving Metadata or triggering Metadata responses. Separate credentials may be required for each uCDN and Domain combination from which content redirections may originate.

4. CDNI Metadata Management

The Metadata creation, modification, retrieval and removal protocols are defined in the following sections. All use a simple HTTP-based approach. The protocol, in general, SHOULD be data format agnostic. The examples shown herein use an XML representation for MI requests/responses, however, other well-defined representations (e.g., JSON) are also acceptable. The examples shown illustrate functionality required to support the data model described in Section 2, however, any protocol which allows for the forced retrieval, invalidation, and removal of Metadata could also be acceptable.

Metadata creation/update is distinguished from retrieval by the HTTP method. Metadata creation/update MUST use the POST method. Metadata retrieval MUST use the GET method. Metadata MUST be removed if the value field is empty (i.e., updating the value to be an empty string MUST force removal of the entire Metadata entry and all associated Base Address entries).

A trigger API is also specified to initiate retrieval of Metadata. The uCDN may issue a trigger to the dCDN to force (re)acquisition of Metadata by the dCDN. The trigger API MUST use the POST method.

In addition to being secured using SSL with client authentication as described in the Security Considerations section, the MI SHOULD also employ an additional Agent authentication mechanism to filter requests and results. In the examples shown below, HTTP basic authentication is used for Agent authentication, though other methods (e.g., HTTP digest authentication or URL hashing) could also be used.

4.1. Metadata API

The Metadata for a Domain is created/modified/retrieved using the "/CDNI/MI/metadata" API. The metadata API **REQUIRES** a single query string argument "domain" which specifies the name of the Domain to which the Metadata being created/modified/retrieved belongs. Three additional **OPTIONAL** arguments **MAY** also be provided when retrieving metadata: "name" which specifies the name of the Metadata field to create/modify/retrieve, "uri" which specifies the URI for which the Metadata must apply, and/or "agent" which specifies the agent(s) to which the Metadata is associated, as a comma separated list. The "agent" option **MUST** only be allowed for agents with full read/write permissions.

A simple XML representation of the information provided to the metadata creation/update API or returned from the metadata retrieval API is shown below:

```
<metadatas>
  <metadata>
    <uri></uri>
    <name></name>
    <values>
      <set>
        <value></value>
        <priority></priority>
      </set>
      ...
    </values>
    <must_enforce></must_enforce>
    <ttn></ttn>
    <agent></agent>
    <baseaddrs>
      <baseaddr></baseaddr>
      ...
    </baseaddrs>
  </metadata>
  ...
</metadatas>
```

Metadata retrieval for a Domain may be triggered using the "/CDNI/MI/trigger" API. The trigger API provides the information required to issue a metadata API retrieval request (i.e., the "domain", "name", and "uri" query string arguments). The metadata API **REQUIRES** a single query string argument "action" which specifies what type of action is being triggered.

The following actions **MUST** be supported:

- o refresh: The dCDN MUST retrieve and update all Metadata specified in the trigger.

The following actions are considered OPTIONAL:

- o preposition: The dCDN SHOULD retrieve and update all Metadata specified in the trigger.

A simple XML representation of the information provided to the trigger API is shown below:

```
<triggers>
  <trigger>
    <host></host>
    <domain></domain>
    <name></name>
    <uri></uri>
  </trigger>
  ...
</triggers>
```

4.1.1.1. Metadata Creation

The following example creates three new Metadata "color" for the "dcdn" Agent in the "acme" Domain, issued by the "ucdn" Agent to the uCDN MI:

```
POST /CDNI/MI/metadata?domain=acme HTTP/1.1
Host: ucdn.mi.cdni.example.com
Accept: */*
Authorization: Basic dWNkbjpw4eHg=
Content-Length: 1053
Content-Type: application/x-www-form-urlencoded
```

```
<metadatas>
  <metadata>
    <uri>/grass/*</uri>
    <name>color</name>
    <values>
      <set>
        <value>brown</value>
        <priority>0</priority>
      </set>
    </values>
    <must_enforce>>false</must_enforce>
    <ttl></ttl>
    <agent>dcdn</agent>
    <baseaddr>
```



```

        <baseaddr>*.acme.com</baseaddr>
      </baseaddrs>
    </metadata>
    <metadata>
      <uri>/grass/on/the/other/side/*</uri>
      <name>color</name>
      <values>
        <set>
          <value>green</value>
          <priority>0</priority>
        </set>
      </values>
      <must_enforce>true</must_enforce>
      <ttl></ttl>
      <agent>dcdn</agent>
      <baseaddrs>
        <baseaddr>*.acme.com</baseaddr>
      </baseaddrs>
    </metadata>
    <metadata>
      <uri>/glasses/*</uri>
      <name>color</name>
      <values>
        <set>
          <value>violet</value>
          <priority>0</priority>
        </set>
      </values>
      <must_enforce>false</must_enforce>
      <ttl></ttl>
      <agent>ucdn</agent>
      <baseaddrs>
        <baseaddr>*.acme.com</baseaddr>
      </baseaddrs>
    </metadata>
  </metadatas>

```

4.1.2. Metadata Update

The following example updates the "color" Metadata for the "/glasses/*" portion of the "acme" Domain and "dcdn" Agent, issued by the "ucdn" Agent to the uCDN MI:

```
POST /CDNI/MI/metadata?domain=acme HTTP/1.1
Host: ucdn.mi.cdni.example.com
Accept: */*
Authorization: Basic dWNkbjp4eHg=
Content-Length: 361
Content-Type: application/x-www-form-urlencoded
```

```
<metadatas>
  <metadata>
    <uri>/glasses/*</uri>
    <name>color</name>
    <values>
      <set>
        <value>rose</value>
        <priority>0</priority>
      </set>
      <set>
        <value>violet</value>
        <priority>2</priority>
      </set>
    </values>
    <must_enforce>true</must_enforce>
    <ttl></ttl>
    <agent>ucdn</agent>
    <baseaddrs>
      <baseaddr>*.acme.com</baseaddr>
    </baseaddrs>
  </metadata>
</metadatas>
```

4.1.3. Metadata Refresh Trigger

The following example triggers the refresh of all "color" Metadata for the "acme" Domain. The trigger is issued by the "ucdn" Agent to the dCDN MI and is intended to force the "dcdn" Agent to retrieve Metadata from the uCDN MI.

```
POST /CDNI/MI/trigger?action=refresh HTTP/1.1
Host: dcdn.mi.cdni.example.com
Accept: */*
Authorization: Basic dWNkbjp4eHg=
Content-Length: 155
Content-Type: application/x-www-form-urlencoded
```

```
<triggers>
  <trigger>
    <host>ucdn.mi.cdni.example.com</host>
    <domain>acme</domain>
    <name>color</name>
    <uri></uri>
  </trigger>
</triggers>
```

The following example triggers the refresh of all Metadata for the URI `/grass/on/this/side`, in the `"acme"` Domain. The trigger is issued by the `"ucdn"` Agent to the `dCDN` MI and is intended to force the `"dcdn"` Agent to retrieve Metadata from the `uCDN` MI.

```
POST /CDNI/MI/trigger?action=refresh HTTP/1.1
Host: dcdn.mi.cdni.example.com
Accept: */*
Authorization: Basic dWNkbjp4eHg=
Content-Length: 169
Content-Type: application/x-www-form-urlencoded
```

```
<triggers>
  <trigger>
    <host>ucdn.mi.cdni.example.com</host>
    <domain>acme</domain>
    <name></name>
    <uri>/grass/on/this/side</uri>
  </trigger>
</triggers>
```

4.1.4. Metadata Retrieval

The following example retrieves all `"color"` Metadata for the `"acme"` Domain. The request was issued by the `"dcdn"` Agent to the `uCDN` MI, and the results are filtered for the `"dcdn"` Agent:

```
GET /CDNI/MI/metadata?domain=acme&name=color HTTP/1.1
Host: ucdn.mi.cdni.example.com
Accept: */*
Authorization: Basic ZGNkbjp5eXk=

HTTP/1.1 200 OK
Content-Length: 714
Connection: close
Content-Type: text/xml
```

```
<metadatas>
  <metadata>
    <uri>/grass/*</uri>
    <name>color</name>
    <values>
      <set>
        <value>brown</value>
        <priority>0</priority>
      </set>
    </values>
    <must_enforce>>false</must_enforce>
    <ttl></ttl>
    <agent>dcdn</agent>
    <baseaddrs>
      <baseaddr>*.acme.com</baseaddr>
    </baseaddrs>
  </metadata>
  <metadata>
    <uri>/grass/on/the/other/side/*</uri>
    <name>color</name>
    <values>
      <set>
        <value>green</value>
        <priority>0</priority>
      </set>
    </values>
    <must_enforce>true</must_enforce>
    <ttl></ttl>
    <agent>dcdn</agent>
    <baseaddrs>
      <baseaddr>*.acme.com</baseaddr>
    </baseaddrs>
  </metadata>
</metadatas>
```

The following example retrieves the Metadata for the URI "/grass/on/this/side" in the "acme" Domain. The request was issued by and the results are filtered for the "dcdn" Agent:

```
GET /CDNI/MI/metadata?domain=acme&uri=/grass/on/this/side HTTP/1.1
Host: ucdn.mi.cdni.example.com
Accept: */*
Authorization: Basic ZGNkbjp5eXk=

HTTP/1.1 200 OK
Content-Length: 361
Connection: close
Content-Type: text/xml
```

```
<metadatas>
  <metadata>
    <uri>/grass/*</uri>
    <name>color</name>
    <values>
      <set>
        <value>brown</value>
        <priority>0</priority>
      </set>
    </values>
    <must_enforce>>false</must_enforce>
    <ttl></ttl>
    <agent>dcdn</agent>
    <baseaddrs>
      <baseaddr>*.acme.com</baseaddr>
    </baseaddrs>
  </metadata>
</metadatas>
```

4.1.1.5. Metadata Removal

The following example removes the violet "color" Metadata value for the URI "/glasses/*" and the "ucdn" Agent in the "acme" Domain by setting the value to an empty string, issued by the "ucdn" Agent to the uCDN MI:

```
POST /CDNI/MI/metadata?domain=acme HTTP/1.1
Host: ucdn.mi.cdni.example.com
Accept: */*
Authorization: Basic dWNkbjp4eHg=
Content-Length: 225
Content-Type: application/x-www-form-urlencoded
```

```
<metadatas>
  <metadata>
    <uri>/glasses/*</uri>
    <name>color</name>
    <values>
      <set>
        <value/>
        <priority>2</priority>
      </set>
    </values>
    <agent>ucdn</agent>
  </metadata>
</metadatas>
```

4.1.6. Metadata Errors

For any update, retrieval, or trigger request with malformed XML, the MI SHOULD respond with a 400 Bad Request status code. Ancillary unknown tags MAY be ignored.

For any trigger requests with an unsupported action, the MI SHOULD respond with a 403 Forbidden status code.

For any update or retrieval request for a uri/name/domain_id tuple which does not exist, the MI SHOULD respond with a 404 Not Found status code.

For any request which lacks a valid Agent authorization, the MI MUST respond with a 401 Unauthorized status code. This includes Agents with valid credentials, but who are marked as read_only and have requested Metadata associated with an alternate Agent through the specification of an "agent" query string parameter.

For any request which results in Metadata with an expired TTL, and for which an update cannot be retrieved from an upstream MI, the MI MUST respond to with a 500 Internal Server status code.

4.1.7. Metadata Prepositioning

The metadata creation/modification/removal APIs discussed above SHOULD only be used by uCDNs to manage Metadata in the local CDN.

Though the metadata creation/modification/removal APIs could be used to preposition metadata in dCDNs, the trigger API allows the uCDN to force refresh of the dCDN Metadata without directly posting Metadata to the dCDN. This allows the dCDNs to manage retrieval of Metadata using lazy updates.

dCDNs SHOULD NOT modify metadata dictated by a uCDN. dCDNs SHOULD only be assigned Agents with read_only access and SHOULD NOT have access to uCDN Domain or Agent APIs (restricted through the use of different SSL client authentication certificates, as described in the Security Considerations section).

5. Metadata Definitions

This section defines a base set of Metadata which SHOULD be supported by all CDNI implementations.

5.1. Origin Server

Content which is not pre-positioned must be acquired by the CDN from an origin server. The origin server Metadata specifies the base URL to which the content request URI may be appended in order to acquire the content. The origin server Metadata is defined as having the name "origin_server", with valid values containing a comma separated list of base URLs, and the must_enforce flag set to false:

```
name: origin_server
value: <url>
must_enforce: false
```

In some cases, multiple non-load balanced origin servers may be available for content acquisition. The origin server Metadata SHOULD support an unprioritized comma separate list of base URL values.

Note: The origin list Metadata is not a must_enforce, since, if the content cannot be acquired, there is no threat of unauthorized content distribution. Other Metadata or content pre-positioning may negate the need for origin server Metadata.

5.2. Activation Time

Content may be pre-positioned in anticipation of demand, however, the content license may have restrictions on delivery timeframe. The activation time Metadata specifies the first time at which the content may be delivered. The activation time Metadata is defined as having the name "activation_time", with valid timestamp values that MUST conform to RFC3339 [RFC3339], and the must_enforce flag set to

true:

```
name: activation_time
value: <timestamp>
must_enforce: true
```

If the activation time Metadata is set and the current time is less than the specified activation time, the CDN MUST respond to requests for that content with a 403 Forbidden status code (or equivalent for the given non-HTTP request protocol).

5.3. Deactivation Time

Content may be pre-positioned in anticipation of demand, however, the content license may have restrictions on delivery timeframe. The deactivation time Metadata specifies the last time at which the content may be delivered. The deactivation time Metadata is defined as having the name "deactivation_time", with valid timestamp values that MUST conform to RFC3339 [RFC3339], and the must_enforce flag set to true:

```
name: deactivation_time
value: <timestamp>
must_enforce: true
```

If the deactivation time Metadata is set and the current time is greater than the specified activation time, the CDN MUST respond to requests for that content with a 403 Forbidden status code (or equivalent for the given non-HTTP request protocol).

5.4. Administrative Disable

It is sometimes necessary to temporarily disable the distribution of certain media (e.g., inappropriate content, irregular access patterns, etc.) within a set accessibility period (i.e., the activation/deactivation time range). The administrative disable Metadata instructs the CDN not to deliver the specified content under any circumstances. The administrative disable Metadata is defined as having the name "admin_disable", with two valid values "true" and "false", and the must_enforce flag set to true:

```
name: admin_disable
value: [true | false]
must_enforce: true
```

If the administrative disable Metadata is set to "true", the CDN MUST respond to requests for that content with a 403 Forbidden status code (or equivalent for the given non-HTTP request protocol).

5.5. Delegation Depth

CSPs may wish to prevent cascading CDNs to enforce licensing restrictions. The delegation depth Metadata instructs the CDN to only delegate requests for the specified content if the delegation depth is greater than zero. If the depth is less than or equal to zero, a uCDN should not delegate requests for the specified content to any dCDNs under any circumstances. When distributing the delegation depth Metadata the uCDN MUST decrement the value of delegation depth by at least one if the current value is greater than zero. The uCDN MAY choose not to decrement the value if the value is already less than or equal to zero. The uCDN MAY decrement by more than one in order to get to zero. The delegation depth Metadata is defined as having the name "delegate_depth", with an integer value and the must_enforce flag set to true:

```
name: delegate_depth
value: <integer>
must_enforce: true
```

If the delegation depth Metadata is less than or equal to 0, the CDN MUST either service the content requests itself or respond to requests for that content with a 504 Server Busy status code (or equivalent for the given non-HTTP request protocol).

5.6. Footprint Filter

CSPs often purchase rights to content which are only valid when accessed from certain locations (e.g., within a given country or through a given access network). The footprint filter Metadata provides a list of valid source IP subnets from which content requests may be accepted. The footprint filter Metadata is defined as having the name "footprint", with valid values containing a comma separated list of IP subnet definitions, and the must_enforce flag set to true:

```
name: footprint
value: <ip_subnet> [, <ip_subnet>]...
must_enforce: true
```

If the footprint filter Metadata is set and the source address of a requesting client does not match any of the IP subnets listed, the CDN MUST respond to the content request with a 403 Forbidden status code (or equivalent for the given non-HTTP request protocol).

5.7. HTTP Header Filter

CSPs often desire the ability to filter requests based on the existence of specific HTTP header fields and values (e.g., User-Agent headers for device detection or custom headers inserted by client-side applications). The HTTP header filter Metadata provides a list of HTTP header names and values which MUST be verified. The HTTP header filter Metadata is defined as having the name "http_filter_headers", with valid values containing a comma separated list of HTTP header names and regular expression matching criteria definitions, and the must_enforce flag set to true:

```
name: http_filter_headers
value: <name>:<regex> [, <name>:<regex>]...
must_enforce: true
```

If the HTTP header filter Metadata is set and the HTTP headers of the content request do not match all of the filters specified, the CDN MUST respond to the content request with a 403 Forbidden status code (or equivalent for the given non-HTTP request protocol).

5.8. HTTP Header Logging

CSP client applications often include proprietary headers in their content requests (e.g., for user tracking or analytics collection) which may be needed for business reasons (e.g., billing) or may be useful for debugging purposes. The HTTP header logging Metadata provides a list of HTTP header names whose values MUST be extracted and logged with the normal per-request information passed through the CDNI logging interface. The HTTP header logging Metadata is defined as having the name "http_logging_headers", with valid values containing a comma separated list of HTTP header names, and the must_enforce flag optionally set to true (depending on the application):

```
name: http_logging_headers
value: <name> [, <name>]...
must_enforce: [true | false]
```

If the HTTP header logging Metadata is set and the content request contains HTTP headers which match any of the header names listed, the CDN MUST extract all matching headers and add them to the per-request log message.

5.9. Protocol Filter

Though content is typically only accessible using specific a protocol (e.g., HTTP, RTMP, or RTSP), a CSP may wish to explicitly allow/

disallow access to certain content for a given protocol. The protocol filter Metadata provides a list of allowed protocols via which content may be delivered. The protocol filter Metadata is defined as having the name "protocol", with valid values containing a comma separate list of protocol strings, and the must_enforce flag set to true:

```
name: protocols
value: <protocol> [, <protocol>]...
must_enforce: true
```

If the protocol filter Metadata is set and the request protocol does not match any protocol in the list, the CDN MUST respond to the content request with a 403 Forbidden status code (or equivalent for the given non-HTTP request protocol).

5.10. SSL Required

CSPs which require delivery privacy may require dCDNs to support the same SSL configurations which were applied to the uCDN. The SSL required Metadata expresses the requirement to enforce SSL on content request connections and provides the necessary key and certificate information required for server authentication. The SSL required Metadata is defined as having the name "ssl_required", with valid values containing two URLs (comma separated) which point to the key and certificate, respectively, and the must_enforce flag set to true:

```
name: ssl_required
value: <key_url>,<cert_url>
must_enforce: true
```

If the SSL required Metadata is set and the request is not received over an SSL channel, the CDN MUST respond to the content request with a 403 Forbidden status code (or equivalent for the given non-HTTP request protocol).

Note: Retrieval of server key and certificate information SHOULD be performed in a secure manner. Retrieval could be implemented through the CDNI MI, however, this is not required.

5.11. SSL Client Authentication Required

CSPs which require client authentication may require dCDNs to support a SSL client authentication configuration which was applied to the uCDN. The SSL client authentication required Metadata expresses the requirement to enforce SSL client authentication on content requests and provides the necessary certificate authority (CA) information for authenticating clients. The SSL client authentication required

Metadata is defined as having the name "ssl_auth_required", with valid values containing a single URL which points to the CA certificate to be used in client verification, and the must_enforce flag set to true:

```
name: ssl_auth_required
value: <ca_url>
must_enforce: true
```

If the SSL client authentication required Metadata is set and the client certificate cannot be verified using the CA certificate, the CDN MUST respond with a handshake_failure alert.

5.12. URL Hash

TBD.

[Ed. Note: There are many proprietary URL hashing techniques in use today with varying timestamp formats, query string parameter names, hashing algorithm combinations, etc. A generic definition of URL hashing algorithm parameters, capable of supporting all algorithms would be best. An alternative of defining specific algorithms and assigning each an enumerated identifier would also work.]

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

There are a number of security concerns associated with the MI as Metadata may be used to influence CDNI request routing. Metadata may describe content acquisition parameters or content security restrictions. Altering Metadata or inhibiting Metadata discovery may impact content distribution. Some MI concerns include:

- o intercepting and discarding Metadata requests to prevent content acquisition may be used as a denial of service attack,
- o altering content acquisition Metadata to prevent content acquisition may be used as a denial of service attack, and
- o spoofing content security Metadata to disable delivery restrictions may be used to circumvent rights management.

To combat these concerns, unauthorized access to the MI MUST be

prevented. The use of SSL with client authentication SHOULD be used for all MI APIs. Deployments in controlled environments where physical security and IP address white-listing is employed MAY choose not to use SSL. Different client authentication certificates SHOULD be used to protect access to Domain and Agent APIs, as well as uCDN access to the Metadata API, differently from dCDN access to the Metadata API. Deployments where uCDNs and dCDNs are mutually trusted entities (e.g., when uCDNs and dCDNs are controlled by the same corporate organization) MAY choose to use a single client authentication certificate.

8. Acknowledgements

The authors would like to thank Daniel Biagini, Susan He, Francois Le Faucheur, Kent Leung, Ben Niven-Jenkins, Gilles Bertrand, and Raj Nair for their helpful reviews and comments.

9. Appendix A: Domain API

Domain creation, modification, retrieval, and removal protocols are defined in the following sections. All use a simple HTTP-based approach. The protocol, in general, SHOULD be data format agnostic. The examples shown herein use an XML representation for MI requests/responses, however, other well-defined representations (e.g., JSON) are also acceptable. The examples shown illustrate the functionality required to support the data model described in Section 2, however, any protocol which allows for the creation, modification, retrieval, and removal of Domains could also be acceptable.

Domain creation/update is distinguished from domain retrieval and removal by the HTTP method. Domain creation/update MUST use the POST method. Domain retrieval MUST use the GET method. Domain removal MUST use the DELETE method.

All Agents and Metadata MUST be associated with a Domain. A Domain is created/modified/retrieved/removed using the "/CDNI/MI/domain" API. The domain API REQUIRES a single query string argument "domain" which specifies the name of the Domain to be created/modified/retrieved.

A simple XML representation of the information provided to the domain creation/update API or returned from the domain retrieval API is shown below:

```
<domain>
  <provider></provider>
  <description></description>
</domain>
```

9.1. Domain Creation

The following example creates a new Domain "acme":

```
POST /CDNI/MI/domain?domain=acme HTTP/1.1
Host: host.mi.cdni.example.com
Accept: */*
Content-Length: 81
Content-Type: application/x-www-form-urlencoded

<domain>
  <provider>acme</provider>
  <description>acme</description>
</domain>
```

9.2. Domain Update

The following example updates the "acme" Domain:

```
POST /CDNI/MI/domain?domain=acme HTTP/1.1
Host: host.mi.cdni.example.com
Accept: */*
Content-Length: 209
Content-Type: application/x-www-form-urlencoded

<domain>
  <provider>acme rocket-powered products, inc</provider>
  <description>fine purveyors of high quality anvils, rubber bands,
    bird seed, and rocket-powered footwear.</description>
</domain>
```

9.3. Domain Retrieval

The following example retrieves the updated "acme" Domain information:

```
GET /CDNI/MI/domain?domain=acme HTTP/1.1
Host: host.mi.cdni.example.com
Accept: */*
```

```
HTTP/1.1 200 OK
Content-Length: 209
Connection: close
Content-Type: text/xml
```

```
<domain>
  <provider>acme rocket-powered products, inc</provider>
  <description>fine purveyors of high quality anvils, rubber bands,
    bird seed, and rocket powered footwear</description>
</domain>
```

The MI MAY support bulk retrieval of Domains through the use of a comma separated list of Domain names in the domain query string parameter.

9.4. Domain Removal

The following example removes the "acme" Domain:

```
DELETE /CDNI/MI/domain?domain=acme HTTP/1.1
Host: host.mi.cdni.example.com
Accept: */*
```

9.5. Domain Errors

Any update or retrieval request with malformed XML SHOULD respond with a 400 Bad Request status code. Ancillary unknown tags MAY be ignored.

Any update or retrieval request for a Domain which does not exist SHOULD respond with a 404 Not Found status code.

10. Appendix B: Agent API

Agent creation, modification, retrieval, and removal protocols are defined in the following sections. All use a simple HTTP-based approach. The protocol, in general, SHOULD be data format agnostic. The examples shown herein use an XML representation for MI requests/responses, however, other well-defined representations (e.g., JSON) are also acceptable. The examples shown illustrate the functionality required to support the data model described in Section 2, however, any protocol which allows for the creation, modification, retrieval,

and removal of Agents could also be acceptable.

Agent creation/update is distinguished from Agent retrieval and removal by the HTTP method. Agent creation/update MUST use the POST method. Agent retrieval MUST use the GET method. Agent removal MUST use the DELETE method and specify the Agent name(s) in the query string.

All Metadata MUST be associated with an Agent. An Agent is created/modified/retrieved/removed using the "/CDNI/MI/agent" API. The agent API REQUIRES a single query string argument "domain" which specifies the name of the Domain to which the Agent has access. In the case of DELETES, the agent API also REQUIRES a query string argument "agent" which specifies the name(s) of the Agent(s) to remove, as a comma separated list.

A simple XML representation of the information provided to the agent creation/update API or returned from the agent retrieval API is shown below:

```
<agents>
  <agent>
    <username></username>
    <password></password>
    <read_only></read_only>
  </agent>
  ...
</agents>
```

10.1. Agent Creation

The following example creates three new Agents "ucdn", "dcdn1", and "dcdn2" for the "acme" Domain:


```
POST /CDNI/MI/agent?domain=acme HTTP/1.1
Host: host.mi.cdni.example.com
Accept: */*
Content-Length: 362
Content-Type: application/x-www-form-urlencoded
```

```
<agents>
  <agent>
    <username>ucdn</username>
    <password>xxx</password>
    <read_only>>false</read_only>
  </agent>
  <agent>
    <username>dcdn1</username>
    <password>aaa</password>
    <read_only>>false</read_only>
  </agent>
  <agent>
    <username>dcdn2</username>
    <password>bbb</password>
    <read_only>>false</read_only>
  </agent>
</agents>
```

10.2. Agent Update

The following example updates the "dcdn1" and "dcdn2" Agents in the "acme" Domain:

```
POST /CDNI/MI/agent?domain=acme HTTP/1.1
Host: host.mi.cdni.example.com
Accept: */*
Content-Length: 245
Content-Type: application/x-www-form-urlencoded
```

```
<agents>
  <agent>
    <username>dcdn1</username>
    <password>yyy</password>
    <read_only>>true</read_only>
  </agent>
  <agent>
    <username>dcdn2</username>
    <password>zzz</password>
    <read_only>>true</read_only>
  </agent>
</agents>
```

10.3. Agent Retrieval

The following example retrieves the updated Agent information for the "acme" Domain:

```
GET /CDNI/MI/agent?domain=acme HTTP/1.1
Host: host.mi.cdni.example.com
Accept: */*
```

```
HTTP/1.1 200 OK
Content-Length: 360
Connection: close
Content-Type: text/xml
```

```
<agents>
  <agent>
    <username>ucdn</username>
    <password>xxx</password>
    <read_only>false</read_only>
  </agent>
  <agent>
    <username>dcdn1</username>
    <password>yyy</password>
    <read_only>true</read_only>
  </agent>
  <agent>
    <username>dcdn2</username>
    <password>zzz</password>
    <read_only>true</read_only>
  </agent>
</agents>
```

10.4. Agent Removal

The following example removes the "dcdn1" Agent from the "acme" Domain:

```
DELETE /CDNI/MI/agent?domain=acme&agent=dcdn1 HTTP/1.1
Host: host.mi.cdni.example.com
Accept: */*
```

10.5. Agent Errors

Any update or retrieval request with malformed XML SHOULD respond with a 400 Bad Request status code. Ancillary unknown tags MAY be ignored.

Any update or retrieval requests for an Agent which does not exist SHOULD respond with a 404 Not Found status code.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.

11.2. Informative References

- [I-D.davie-cdni-framework]
Davie, B., Ed. and L. Peterson, Ed., "Framework for CDN Interconnection draft-davie-cdni-framework-01", October 2011.
- [I-D.ietf-cdni-requirements]
Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements draft-ietf-cdni-requirements-02", December 2011.
- [I-D.ietf-cdni-use-cases]
Bertrand, G., Stephan, E., Watson, G., Burbridge, T., Eardley, P., and K. Ma, "Use Cases for Content Delivery Network Interconnection draft-ietf-cdni-use-cases-04", March 2012.

Author's Address

Kevin J. Ma
Azuki Systems, Inc.
43 Nagog Park
Acton, MA 01720
USA

Phone: +1 978-844-5100
Email: kevin.ma@azukisystems.com

Content Delivery Networks
Interconnection
Internet-Draft
Intended status: Informational
Expires: January 17, 2013

J. Seedorf
NEC
July 16, 2012

CDNI Request Routing with ALTO
draft-seedorf-cdni-request-routing-alto-02

Abstract

Network Service Providers (NSPs) are currently considering to deploy Content Delivery Networks (CDNs) within their networks. As a consequence of this development, there is a need for interconnecting these local CDNs. The necessary interfaces for inter-connecting CDNs are currently being defined in the Content Delivery Networks Interconnection (CDNI) WG. This document focusses on the Request Routing Interface of CDNI, and more specifically on how the solutions currently being defined in the Application Layer Traffic Optimization (ALTO) WG can improve CDNI request routing. The overall intention behind this document is to foster discussions (in the CDNI as well as in the ALTO WG) regarding if, how, and under what conditions ALTO can be useful to optimize CDNI request routing. As basis for this discussion, this document provides concrete examples of how ALTO can be integrated within CDNI request routing and in particular in the process of selecting a downstream CDN. The examples in this document are based on the use cases and examples currently being discussed in the CDNI WG.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 17, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. ALTO within CDNI Request Routing	4
3. Assumptions and High-Level Design Considerations	6
4. Selection of a Downstream CDN with ALTO	8
4.1. Footprint Advertisement with ALTO Network Map	8
4.2. Using ALTO maps to convey Additional Information for Downstream CDN Selection	8
4.3. Example of Selecting a Downstream CDN based on ALTO Maps	10
4.4. Advantages of using ALTO	11
5. Useful ALTO extensions for CDNI Request Routing	13
6. Security Considerations	15
7. Summary and Outlook	16
8. Acknowledgements	17
9. Informative References	18
Author's Address	20

1. Introduction

Many Network Service Providers (NSPs) are currently considering or have already started to deploy Content Delivery Networks (CDNs) within their networks. As a consequence of this development, there is a need for interconnecting these local CDNs. Content Delivery Networks Interconnection (CDNI) has the goal of standardizing protocols to enable such interconnection of CDNs [I-D.ietf-cdni-problem-statement].

The CDNI problem statement envisions four interfaces to be standardized within the IETF for CDN interconnection [I-D.ietf-cdni-problem-statement]:

- o CDNI Request Routing Interface
- o CDNI Metadata Interface
- o CDNI Logging Interface
- o CDNI Control Interface

This document focusses solely on the CDNI Request Routing Interface. In particular, this document shows concrete examples of how ALTO [RFC5693] can be integrated in CDNI request routing. The goal of this document is to show in what cases ALTO can benefit CDNI request routing, giving concrete examples and explaining how ALTO improves CDNI request routing in each of these examples. The examples used in this document are based on the use cases and request routing proposals currently being discussed in the CDNI WG [I-D.ietf-cdni-use-cases] [I-D.peterson-CDNI-strawman] and in the ALTO WG [I-D.jenkins-alto-cdn-use-cases]. The overall rationale of this document is to foster discussions (in the CDNI as well as in the ALTO WG) regarding if, how, and under what conditions ALTO can be useful to optimize CDNI request routing. Most importantly, the document has the goal of finding consensus regarding which part of the CDNI request routing interface can use ALTO.

A previous version of this document [I-D.seedorf-alto-for-cdni] contained detailed examples of actual request routing and surrogate selection with ALTO. This version solely focuses on selection of a downstream CDN and how ALTO can support such downstream CDN selection.

Throughout this document, we use the terminology for CDNI defined in [I-D.ietf-cdni-problem-statement].

2. ALTO within CDNI Request Routing

The main purpose of the CDNI Request Routing Interface is described in [I-D.ietf-cdni-problem-statement] as follows: "The CDNI Request Routing interface enables a Request Routing function in an upstream CDN to query a Request Routing function in a downstream CDN to determine if the downstream CDN is able (and willing) to accept the delegated content request and to allow the downstream CDN to control what the upstream Request Routing function should return to the User Agent in the redirection message". On a high level, the scope of the CDNI Request Routing Interface therefore contains two main tasks:

- o A) Determining if the downstream CDN is willing to accept a delegated content request
- o B) Redirecting the content request coming from an upstream CDN to the proper entry point or entity in the downstream CDN

More precisely, in [I-D.ietf-cdni-framework] the request routing interface is broadly divided into two functionalities:

- o 1) the asynchronous advertisement of footprint and capabilities by a dCDN that allows a uCDN to decide whether to redirect particular user requests to that dCDN;
- o 2) the synchronous operation of actually redirecting a user request.

According to consensus found at the CDNI working group session at IETF-82, we refer to 1) as "Request Routing Interface - Footprint and Capabilities Advertisement" and 2) as "Request Routing Interface - Redirection" in this document. A previous version of this document [I-D.seedorf-alto-for-cdni] provided some concrete examples how ALTO could be used for the actual "redirection" part of the request routing interface. Based on feedback received from the CDNI working group (mostly at the IETF-82 meeting), this document solely focuses on the "Footprint and Capabilities Advertisement" part of the request routing interface. In particular, the scope of the current version of this document is to show how ALTO [RFC5693] can be used for selecting a downstream CDN. Thus, the scope of the current document is to provide examples and discuss how a downstream CDN can advertise its footprint and other information by means of ALTO.

Application Layer Traffic Optimization (ALTO) is an approach for guiding the resource provider selection process in distributed applications that can choose among several candidate resources providers to retrieve a given resource. By conveying network layer (topology) information, an ALTO server can provide important

information to "guide" the resource provider selection process in distributed applications. Usually, it is assumed that an ALTO server conveys information these applications cannot measure themselves [RFC5693].

Originally, ALTO was motivated by the huge amount of cross-ISP traffic generated by P2P applications [RFC5693]. Recently, however, ALTO is also being considered for improving the request routing in CDNs [I-D.jenkins-alto-cdn-use-cases]. In this context, it has also been proposed to use ALTO for selecting an entry-point in a downstream NSP's network (see section 3.4 "CDN delivering Over-The-Top of a NSP's network" in [I-D.jenkins-alto-cdn-use-cases]). Also, the CDNI problem statement explicitly mentions ALTO as a candidate protocol for "algorithms for selection of CDN or Surrogate by Request-Routing systems" [I-D.ietf-cdni-problem-statement]. Yet, there have not been concrete proposals so far on how to use ALTO in the context of CDN interconnection. This document tries to close this gap by giving some examples on how ALTO could be used within CDNI request routing.

3. Assumptions and High-Level Design Considerations

In this section we list some assumptions and design issues to be considered when using ALTO for CDNI "footprint and capabilities advertisement":

- o As explicitly being out-of-scope for CDNI [I-D.ietf-cdni-problem-statement], the examples used in this document assume that ingestion of content or acquiring content across CDNs is not part of request routing as considered within CDNI standardization work. The focus of using ALTO (as considered in this document) is hence on request routing only, assuming that the content (desired by the end user) is available in the downstream CDN (or can be acquired by the downstream CDN by some means).
- o Federation Model: "footprint and capabilities advertisement" and in general CDN request routing depends on the federation model among the CDN providers. Designing a suitable solution thus depends on whether a solution is needed for different settings, where CDNs consist of both NSP CDNs (serving individual ASes) and general, traditional CDNs (such as Akamai). We assume that CDNI is not designed for a setting where only NSP CDNs each serve a single AS only.
- o Many CDNs can claim that they can serve any host on the Internet due to Internet connectivity. For example, even if an NSP CDN A does not have surrogate servers inside network C, A can legitimately claim that it can serve customers inside network C. Hence, what a downstream CDN should inform an upstream CDN is performance and capability, and not (only) reachability or coverage. Although one may turn performance into (binary) reachability by defining a threshold, the metric can be content-dependent (image, video, files) or artificial. The requirement of conveying performance and capability is consistent with the context: after all, the foundation of the CDN business is to improve user QoE.
- o In this document, we assume that the upstream CDN (uCDN) makes the decision on selecting a downstream CDN, based on information that each downstream CDN has made available to the upstream CDN. Further, we assume that in principle more than one dCDN may be suitable for a given end-user request (i.e. different dCDNs may claim "overlapping" footprints). The uCDN hence potentially has to select among several candidate downstream CDNs for a given end user request.

- o The term "footprint" has not been precisely defined by the CDNI working group yet [I-D.spp-cdni-rr-foot-cap-semantics]. In this document we assume that some notion of IP prefix-range is suitable to define a footprint of a downstream CDN. Thus, a dCDN footprint may be expressed as an IP-prefix range that a given dCDN claims it can cover. However, in principle any host can reach any other host on the Internet. Thus, simple "coverage" of IP-prefix ranges seems insufficient for a uCDN to make a choice of a dCDN (see [I-D.spp-cdni-rr-foot-cap-semantics] for a discussion of this issue). The uCDN needs additional information that is "tagged along" (either implicitly or explicitly) with such a coverage footprint. Such additional information has the purpose of conveying to the uCDN more information about a footprint, so that the uCDN can judge/assess the delivery quality that is associated with a given dCDN footprint (or part of that footprint, in the likely case that the delivery quality is not the same for a whole dCDN footprint).
- o It is not clear what kind(s) of business, contract, and operational relationships two peering CDNs may form. For the Internet, we see provider-customer and peering as two main relations; providers may use different charging models (e.g., 95-percentile, total volume) and may provide different SLAs. Given such unknown characteristics of CDN peering business agreements, we should design the protocol to support as much diverse potential business and operational models as possible.

4. Selection of a Downstream CDN with ALTO

Under the considerations stated in Section 3, ALTO can help the upstream CDN provider to select a proper downstream CDN provider for a given end user request as follows: Each downstream CDN provider hosts an ALTO server which provides ALTO information (i.e. ALTO network maps and ALTO cost maps [I-D.ietf-alto-protocol]) to an ALTO client at the upstream CDN provider. A network map provided by each of several candidate downstream CDNs can provide information to the upstream CDN provider about each dCDN's "coverage" footprint, e.g. regarding geographical coverage, the (exact or rough) location of "surrogates", the IP-prefix ranges the dCDN claims it can "cover" with "good" delivery quality, or similar. Additional ALTO network maps or cost maps can provide an upstream CDN provider additional information about the footprint each individual dCDN offers, e.g. the "cost" or quality associated with delivering certain content via the downstream CDN which provided such a map. "Cost" in this context is a generic term; many types of costs are possible and can be useful in the context of CDNI request routing (see Section 4.2 for a detailed discussion), e.g. average link load, expected delay, or monetary costs.

4.1. Footprint Advertisement with ALTO Network Map

An ALTO network map contains a "set of Network Location groupings" [I-D.ietf-alto-protocol]. The groupings are defined in the form of so-called "PIDs". A PID is an identifier to group network location endpoints, e.g. IP-addresses in the form of prefixes (see section 4 in [I-D.ietf-alto-protocol] for details).

The concept of an ALTO network map (and the PIDs contained therein) is a natural and straightforward candidate for CDNI footprint advertisement: The downstream CDN provider groups the IP-addresses in its footprint into PIDs and makes these groupings available to an upstream CDN via an ALTO network map. With such a network map, the upstream CDN provider can easily match a given end user request with the footprint of the downstream CDN provider to see if a given downstream CDN can in principle provide "coverage" for the IP-address of the end user. Whenever the footprint changes, the downstream CDN creates an updated network map and makes it available via its ALTO server.

4.2. Using ALTO maps to convey Additional Information for Downstream CDN Selection

Additional information (so that the uCDN can judge/assess the delivery quality that is associated with a given dCDN footprint it received in a network map from the dCDN) can be conveyed to a uCDN with

additional ALTO network maps or ALTO cost maps that the dCDN will provide via its ALTO server. An ALTO cost map contains costs between defined groupings of a corresponding network map (i.e. costs between PIDs): "An ALTO Cost Map defines Path Costs pairwise amongst sets of source and destination Network Locations" [I-D.ietf-alto-protocol]. This concept enables the provider of a cost map to express (and quantify) preferences of a destination network location with respect to a given source network location.

In the context of CDNI, the ALTO cost map concept is an extensive tool to facilitate selection of the "best" downstream CDN because it enables the upstream CDN provider to assess a candidate downstream CDN based on other factors besides simply network coverage (coverage footprint). Most importantly, the cost map concept provides a means for a downstream CDN provider to convey a multitude of dynamically changing information which the upstream CDN provider cannot measure itself (or only roughly estimate) otherwise.

For instance, the following types of "delivery cost" can be conveyed by a downstream CDN provider via ALTO for each combination of source PID and destination PID:

- o Latency: the expected/average RTT
- o Bandwidth: the maximum bandwidth (e.g. due too bottlenecks)
- o Monetary Costs: The amount of actual monetary costs the downstream CDN provider would charge for the delivery of content to a given destination (see also [I-D.liu-cdni-cost])

Normally, an ALTO cost map defines "costs" pairwise among two PIDs [I-D.ietf-alto-protocol]. In the current scope of the CDNI working group, however, the destination of a request routing redirection will always be the request router of the selected downstream CDN (as direct redirection to dCDN surrogates is currently out of scope of the CDNI work). The destination PID in an ALTO cost map offered by a dCDN is thus always (i.e. in all entries) the same. Moreover, this destination PID semantically refers to the whole dCDN (or to its request router, as the decision where to route within the dCDN is outside the scope of the CDNI request routing interface). Therefore, a corresponding ALTO network map for footprint coverage offered by a dCDN should always contain a special PID that covers the whole footprint of the dCDN, i.e. the overall, whole prefix range the dCDN claims it can cover (obviously, it can additionally contain smaller prefix ranges in other PIDs, so that a cost map can have pairwise costs entries of these smaller-scale source PIDs to the overall dCDN coverage PID).

Note that such ALTO cost maps are always of the type N-to-1, i.e. "costs" are expressed for each of N end user source PIDs to 1 single dCDN request router PID. Semantically, the source PID in a CDNI ALTO cost map is thus the end user location, whereas the destination is the request router to which the uCDN redirects the end user request. Note that this perspective is driven by the CDNI request routing. An alternative way - seen from the perspective of content retrieval - would be to have a 1-to-N cost map where the source is always the dCDN and the destination is the end user (with the semantic "if the source dCDN would deliver content to an end user in the destination PID, the costs would be the following").

Alternatively to using cost maps for expressing delivery quality for a given coverage footprint, ALTO networks map could be used. In this case, an additional network map provided by the dCDN groups the dCDN's coverage footprint into several PIDs, where each PID name has a certain "quality" semantic. In other words, all IP-prefixes in a certain PID have the same "quality", and the meaning of this quality is expressed by the PID name.

4.3. Example of Selecting a Downstream CDN based on ALTO Maps

In the following, we will outline an example of dCDN selection by a uCDN based on ALTO maps provided by each dCDN. In the example, an ALTO network map "NM_cov" is used to express the overall "coverage" footprint of each dCDN. In addition (as outlined in Section 4.2), each dCDN provides one or more ALTO cost maps "CM_1", "CM_2", ..., "CM_n" to express the delivery "costs"/quality associated with each PID in the corresponding "NM_cov" coverage footprint network map.

Consider the following example: An upstream CDN (uCDN) has agreed on CDN interconnection with several downstream CDNs (dCDN-a, dCDN-b, and dCDN-c). Each of these downstream CDNs runs an ALTO server to provide information about what locations it can deliver content to (coverage footprint) by means of a network map "NM_cov" and at which "cost" (additional delivery quality information) by means of one or more cost maps "CM_1", "CM_2", ..., "CM_n". uCDN has downloaded from each candidate downstream CDN "NM_cov" and one or more ALTO cost maps (e.g. by using the "Filtered Cost Map" option and different "cost-types" as specified in 7.7.3.2. of [I-D.ietf-alto-protocol]). The ALTO network map provides "coverage" (footprint) for each downstream CDN as aggregated network locations in the form of ALTO PIDs. The cost maps provide the upstream CDN information regarding the delivery quality the selection of each individual downstream CDN would imply depending on the given location of an end user request.

Whenever the upstream CDN receives a request from an end user and has determined that this request is best served by an interconnected

dCDN, the uCDN uses ALTO maps to make a redirection decision. For a given request, assume that only the ALTO network maps provided by dCDN-a and dCDN-c, "NM_cov(dCDN-a)" and "NM_cov(dCDN-c)", indicate that these downstream CDNs can deliver content to the location of the request. In this case, the ALTO costs maps received from dCDN-a and dCDN-c provide useful additional information to the upstream CDN in order to make a selection decision regarding either dCDN-a or dCDN-c. For instance, if both downstream CDNs have provided two ALTO cost maps "CM_monetary" and "CM_latency" - one regarding monetary costs and one regarding expected latency for delivery - uCDN can make a downstream CDN selection based on its preferences: If one downstream CDN can deliver cheaper, but the other faster, ALTO cost maps provide such information in detail to the upstream CDN. This enables the upstream CDN to make a well-considered downstream CDN selection. In particular, the uCDN decision may take into account different delivery quality indicators or other factors (which can be weighted by the upstream CDN to make a decision).

4.4. Advantages of using ALTO

The following reasons make ALTO a suitable candidate protocol for downstream CDN selection as part of CDNI request routing:

- o CDN request routing is done at the application layer. ALTO is a protocol specifically designed to improve application layer traffic (and application layer connections among hosts on the Internet) by providing additional information to applications that these applications could not easily retrieve themselves. For CDNI, this is exactly the case: a uCDN wants to improve application layer CDN request routing by using dedicated information (provided by a dCDN) that the uCDN could not easily obtain otherwise.
- o The semantics of an ALTO network are an exact match for the needed information to convey a footprint by a downstream CDN, in particular if such a footprint is being expressed by IP-prefix ranges.
- o ALTO cost maps are suitable to express various types of delivery "cost" and can hence be used by an upstream to judge the delivery quality associated with a given dCDN for a given end user request. Further, an ALTO cost map can convey relevant network topology information other than simply routing hops or reachability. This facilitates advanced and more sophisticated selection of a downstream CDN based on various metrics by the upstream CDN and increases flexibility to cover different use cases and business models for CDN interconnection.

- o Flexible granularity: The concept of the PID and ALTO network/cost maps allows for different degrees of granularity. This enables a dCDN to differentiate the delivery quality for serving an end user request on a fine granularity depending on the end user location (and not only express delivery quality e.g. on an AS-level). It remains at the discretion of each dCDN how fine-granular the ALTO network and cost maps are that it publishes.
- o ALTO maps can be signed and hence provide inherent integrity protection (see Section 6)

5. Useful ALTO extensions for CDNI Request Routing

It is envisioned that yet-to-be-defined ALTO extensions will be standardized that make the ALTO protocol more suitable and useful for applications other than the originally considered P2P use case [I-D.marocco-alto-next]. Some of these extensions to the ALTO protocol would be useful for ALTO to be used as a protocol within CDNI request routing, and in particular within the "Footprint and Capabilities Advertisement" part of the CDNI request routing interface.

The following proposed extensions to ALTO would be beneficial to facilitate CDNI request routing with ALTO as outlined in Section 4:

- o Server-initiated Notifications and Incremental Updates: In case the footprint or the capabilities of a downstream CDN change abruptly (i.e. unexpectedly from the perspective of an upstream CDN), server initiated notifications would enable a dCDN to directly inform an upstream CDN about such changes. Consider the case where - due to failure - part of the footprint of the dCDN is not functioning, i.e. the CDN cannot serve content to such clients with reasonable QoS. Without server-initiated notifications, the uCDN might still use a very recent network and cost map from dCDN, and therefore redirect request to dCDN which it cannot serve. Similarly, the possibility for incremental updates would enable efficient conveyance of the aforementioned (or similar) status changes by the dCDN to the uCDN. A proposal for server-initiated ALTO updates can be found in [I-D.marocco-alto-ws]. A discussion of incremental ALTO updates can be found in [I-D.schwan-alto-incr-updates].
- o Content Availability on Hosts: A dCDN might want to express CDN capabilities in terms of certain content types (e.g. codecs/formats, or content from certain content providers). A new endpoint property for ALTO that would be able to express such "content availability" would enable a dCDN to make available such information to an upstream CDN. This would enable a uCDN to determine if a given dCDN actually has the capabilities for a given request with respect to the type of content requested.
- o Resource Availability on Hosts or Links: The capabilities on links (e.g. maximum bandwidth) or caches (e.g. average load) might be useful information for an upstream CDN for optimized downstream CDN selection. For instance, if a uCDN receives a streaming request for content with a certain bitrate, it needs to know if it is likely that a dCDN can fulfill such stringent application-level requirements (i.e. can be expected to have enough consistent bandwidth) before it redirects the request. In general, if ALTO

could convey such information via new endpoint properties, it would enable more sophisticated means for downstream CDN selection with ALTO.

6. Security Considerations

One important security consideration is the proper authentication of advertisement information provided by a downstream CDN. The ALTO protocol provides a specification for a signature of ALTO maps (see 8.2.2. of [I-D.ietf-alto-protocol]). ALTO thus provides a proper means for protecting the integrity of footprint advertisement information.

More Security Considerations will be discussed in a future version of this document.

7. Summary and Outlook

This document presented concrete examples of how ALTO can be used within the downstream CDN selection of CDNI Request Routing. Further, the document provides arguments why ALTO is a meaningful protocol in this context. Essentially, ALTO network and cost maps are a means to provide detailed and various types of information to an upstream CDN, in order to facilitate well-considered downstream CDN selection.

The intention of this document is to find consensus in the CDNI WG that ALTO is a useful protocol for CDNI request routing, and that ALTO has many benefits for proper selection of a downstream CDN. The overall objective is to form agreement on how ALTO should be used within the CDNI request routing protocol. It is the intention to capture the outcome of such continuing discussions in future versions of this document.

8. Acknowledgements

Jan Seedorf is partially supported by the COAST project (Content Aware Searching, retrieval and sTreaming, <http://www.coast-fp7.eu>), a research project supported by the European Commission under its 7th Framework Program (contract no. 248036). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the COAST project or the European Commission.

Thanks to Richard Yang for providing valuable comments, and for contributing some design considerations and assumptions.

9. Informative References

- [RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, October 2009.
- [I-D.peterson-CDNI-strawman]
Peterson, L. and J. Hartman, "Content Distribution Network Interconnection (CDNI) Problem Statement", draft-peterson-CDNI-strawman-01 (work in progress), May 2011.
- [I-D.ietf-cdni-problem-statement]
Niven-Jenkins, B., Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", draft-ietf-cdni-problem-statement-08 (work in progress), June 2012.
- [I-D.marocco-alto-next]
Marocco, E. and V. Gurbani, "Extending the Application-Layer Traffic Optimization (ALTO) Protocol", draft-marocco-alto-next-00 (work in progress), January 2012.
- [I-D.ietf-alto-protocol]
Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol", draft-ietf-alto-protocol-12 (work in progress), July 2012.
- [I-D.ietf-cdni-requirements]
Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", draft-ietf-cdni-requirements-03 (work in progress), June 2012.
- [I-D.ietf-cdni-use-cases]
Bertrand, G., Emile, S., Burbridge, T., Eardley, P., Ma, K., and G. Watson, "Use Cases for Content Delivery Network Interconnection", draft-ietf-cdni-use-cases-09 (work in progress), July 2012.
- [I-D.marocco-alto-ws]
Marocco, E. and J. Seedorf, "WebSocket-based server-to-client notifications for the Application-Layer Traffic Optimization (ALTO) Protocol", draft-marocco-alto-ws-01 (work in progress), July 2012.
- [I-D.schwan-alto-incr-updates]
Schwan, N. and B. Roome, "ALTO Incremental Updates",

draft-schwan-alto-incr-updates-02 (work in progress),
July 2012.

[I-D.jenkins-alto-cdn-use-cases]

Niven-Jenkins, B., Watson, G., Bitar, N., Medved, J., and
S. Previdi, "Use Cases for ALTO within CDNs",
draft-jenkins-alto-cdn-use-cases-03 (work in progress),
June 2012.

[I-D.seedorf-alto-for-cdni]

Seedorf, J., "ALTO for CDNI Request Routing",
draft-seedorf-alto-for-cdni-00 (work in progress),
October 2011.

[I-D.ietf-cdni-framework]

Peterson, L. and B. Davie, "Framework for CDN
Interconnection", draft-ietf-cdni-framework-00 (work in
progress), April 2012.

[I-D.liu-cdni-cost]

Liu, H., "A Cost Perspective on Using Multiple CDNs",
draft-liu-cdni-cost-00 (work in progress), October 2011.

[I-D.spp-cdni-rr-foot-cap-semantics]

Seedorf, J., Peterson, J., and S. Previdi, "CDNI Request
Routing: Footprint and Capabilities Semantics",
draft-spp-cdni-rr-foot-cap-semantics-00 (work in
progress), March 2012.

Author's Address

Jan Seedorf
NEC Laboratories Europe, NEC Europe Ltd.
Kurfuersten-Anlage 36
Heidelberg 69115
Germany

Phone: +49 (0) 6221 4342 221
Email: jan.seedorf@neclab.eu
URI: <http://www.neclab.eu>

CDNI	J. Seedorf
Internet-Draft	HFT Stuttgart - Univ. of Applied Sciences
Intended status: Standards Track	Y. Yang
Expires: January 3, 2018	Tongji/Yale
	K. Ma
	Ericsson
	J. Peterson
	Neustar
	July 2, 2017

Content Delivery Network Interconnection (CDNI) Request Routing: CDNI
Footprint and Capabilities Advertisement using ALTO
draft-seedorf-cdni-request-routing-alto-10

Abstract

The Content Delivery Networks Interconnection (CDNI) WG is defining a set of protocols to inter-connect CDNs, to achieve multiple goals such as extending the reach of a given CDN to areas that are not covered by that particular CDN. One component that is needed to achieve the goal of CDNI is the CDNI Request Routing Footprint & Capabilities Advertisement interface (FCI) [RFC7336]. [RFC8008] has defined precisely the semantics of FCI and provided guidelines on the FCI protocol, but the exact protocol is explicitly outside the scope of that document. In this document, we define an FCI protocol using the Application Layer Traffic Optimization (ALTO) protocol.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 3, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Background	4
2.1. Semantics of FCI Advertisement	4
2.2. ALTO Background and Benefits	5
3. CDNI FCI ALTO Service	7
3.1. Server Response Encoding	8
3.1.1. Media Type	8
3.1.2. Meta Information	8
3.1.3. Data Information	8
3.2. Protocol Errors	8
3.3. Examples	8
3.3.1. Basic Example	8
3.3.2. Incremental FCI Update Example	9
3.3.3. FCI Using ALTO Network Map Example	9
4. Security Considerations	9
5. Acknowledgements	10
6. References	10
6.1. Normative References	10
6.2. Informative References	11
Authors' Addresses	11

1. Introduction

Many Network Service Providers (NSPs) are currently considering or have already started to deploy Content Delivery Networks (CDNs) within their networks. As a consequence of this development, there is a need for interconnecting these local CDNs. Content Delivery Networks Interconnection (CDNI) has the goal of standardizing protocols to enable such interconnection of CDNs [RFC6707].

The CDNI problem statement [RFC6707] defines four interfaces to be standardized within the IETF for CDN interconnection:

- o CDNI Request Routing Interface
- o CDNI Metadata Interface
- o CDNI Logging Interface
- o CDNI Control Interface

The main purpose of the CDNI Request Routing Interface is described in [RFC6707] as follows: "The CDNI Request Routing interface enables a Request Routing function in an Upstream CDN to query a Request Routing function in a Downstream CDN to determine if the Downstream CDN is able (and willing) to accept the delegated Content Request. It also allows the Downstream CDN to control what should be returned to the User Agent in the redirection message by the upstream Request Routing function." On a high level, the scope of the CDNI Request Routing Interface therefore contains two main tasks:

- o determining if the downstream CDN is willing to accept a delegated content request;
- o redirecting the content request coming from an upstream CDN to the proper entry point or entity in the downstream CDN.

Correspondingly, the request routing interface is broadly divided into two functionalities:

- o CDNI FCI: the advertisement from a dCDN to a uCDN or a query from a uCDN to a dCDN for the uCDN to decide whether to redirect particular user requests to that dCDN;
- o CDNI RI: the synchronous operation of actually redirecting a user request.

This document focuses solely on CDNI FCI, with a goal to specify a new Application Layer Traffic Optimization (ALTO) [RFC7285] service called 'CDNI/FCI Service', to transport and update CDNI FCI JSON objects, which are defined in a separate document in [RFC8008].

Throughout this document, we use the terminology for CDNI defined in [RFC6707] and [RFC8008].

2. Background

The design of CDNI FCI transport using ALTO depends on understanding of both FCI semantics and ALTO. Hence, we start with a review of both.

2.1. Semantics of FCI Advertisement

The CDNI document on "Footprint and Capabilities Semantics" [RFC8008] defines the semantics for the CDNI FCI. It thus provides guidance on what Footprint and Capabilities mean in a CDNI context and how a protocol solution should in principle look like. The definitions in [RFC8008] depend on [RFC8006]. Here we briefly summarize key related points of [RFC8008] and [RFC8006]. For a detailed discussion, the reader is referred to the RFCs.

- o Footprint and capabilities are tied together and cannot be interpreted independently from each other. In such cases, i.e. where capabilities must be expressed on a per footprint basis, it may be beneficial to combine footprint and capabilities advertisement. [RFC8008] integrates footprint and capabilities with an approach of "capabilities with footprint restrictions".
- o Given that a large part of Footprint and Capabilities Advertisement will actually happen in contractual agreements, the semantics of CDNI Footprint and Capabilities advertisement refer to answering the following question: what exactly still needs to be advertised by the CDNI FCI? For instance, updates about temporal failures of part of a footprint can be useful information to convey via the CDNI request routing interface. Such information would provide updates on information previously agreed in contracts between the participating CDNs. In other words, the CDNI FCI is a means for a dCDN to provide changes/updates regarding a footprint and/or capabilities it has prior agreed to serve in a contract with a uCDN. Hence, server push and incremental encoding will be necessary techniques.
- o Multiple types of footprints are defined in [RFC8006]:
 - * List of ISO Country Codes
 - * List of AS numbers
 - * Set of IP-prefixes

A 'set of IP-prefixes' must be able to contain full IP addresses, i.e., a /32 for IPv4 and a /128 for IPv6, and also IP prefixes with an arbitrary prefix length. There must also be support for

multiple IP address versions, i.e., IPv4 and IPv6, in such a footprint.

- o For all of these mandatory-to-implement footprint types, footprints can be viewed as constraints for delegating requests to a dCDN: A dCDN footprint advertisement tells the uCDN the limitations for delegating a request to the dCDN. For IP prefixes or ASN(s), the footprint signals to the uCDN that it should consider the dCDN a candidate only if the IP address of the request routing source falls within the prefix set (or ASN, respectively). The CDNI specifications do not define how a given uCDN determines what address ranges are in a particular ASN. Similarly, for country codes a uCDN should only consider the dCDN a candidate if it covers the country of the request routing source. The CDNI specifications do not define how a given uCDN determines the country of the request routing source. Multiple footprint constraints are additive, i.e. the advertisement of different types of footprint narrows the dCDN candidacy cumulatively.
- o The following capabilities are defined as 'base' capabilities, i.e. ones that are needed in any case and therefore constitute mandatory capabilities to be supported by the CDNI FCI:
 - * Delivery Protocol (e.g., HTTP vs. RTMP)
 - * Acquisition Protocol (for acquiring content from a uCDN)
 - * Redirection Mode (e.g., DNS Redirection vs. HTTP Redirection as discussed in [RFC7336])
 - * Capabilities related to CDNI Logging (e.g., supported logging mechanisms)
 - * Capabilities related to CDNI Metadata (e.g., authorization algorithms or support for proprietary vendor metadata)

2.2. ALTO Background and Benefits

Application Layer Traffic Optimization (ALTO) [RFC7285] is an approach for guiding the resource provider selection process in distributed applications that can choose among several candidate resources providers to retrieve a given resource. By conveying network layer (topology) information, an ALTO server can provide important information to "guide" the resource provider selection process in distributed applications. Usually, it is assumed that an ALTO server conveys information these applications cannot measure themselves [RFC5693].

Originally, ALTO was motivated by the huge amount of cross-ISP traffic generated by P2P applications [RFC5693]. Recently, however, ALTO is also being considered for improving the request routing in CDNs [I-D.jenkins-alto-cdn-use-cases]. In this context, it has also been proposed to use ALTO for selecting an entry-point in a downstream NSP's network (see section 3.4 "CDN delivering Over-The-Top of a NSP's network" in [I-D.jenkins-alto-cdn-use-cases]). Also, the CDNI problem statement explicitly mentions ALTO as a candidate protocol for "algorithms for selection of CDN or Surrogate by Request-Routing systems" [RFC6707].

The following reasons make ALTO a suitable candidate protocol for downstream CDN selection as part of CDNI request routing and in particular for an FCI protocol:

- o CDN request routing is done at the application layer. ALTO is a protocol specifically designed to improve application layer traffic (and application layer connections among hosts on the Internet) by providing additional information to applications that these applications could not easily retrieve themselves. For CDNI, this is exactly the case: a uCDN wants to improve application layer CDN request routing by using dedicated information (provided by a dCDN) that the uCDN could not easily obtain otherwise.
- o The semantics of an ALTO network map are an exact match for the needed information to convey a footprint by a downstream CDN, in particular if such a footprint is being expressed by IP-prefix ranges.
- o Security: ALTO maps can be signed and hence provide inherent integrity protection (see Section 4)
- o RESTful-Design: The ALTO protocol has undergone extensive revisions in order to provide a RESTful design regarding the client-server interaction specified by the protocol. A CDNI FCI interface based on ALTO would inherit this RESTful design.
- o Error-handling: The ALTO protocol has undergone extensive revisions in order to provide sophisticated error-handling, in particular regarding unexpected cases. A CDNI FCI interface based on ALTO would inherit this thought-through and mature error-handling.
- o Filtered network map: The ALTO Map Filtering Service (see [RFC7285] for details) would allow a uCDN to query only for parts of an ALTO map.

- o Server-initiated Notifications and Incremental Updates: In case the footprint or the capabilities of a downstream CDN change abruptly (i.e. unexpectedly from the perspective of an upstream CDN), server initiated notifications would enable a dCDN to directly inform an upstream CDN about such changes. Consider the case where - due to failure - part of the footprint of the dCDN is not functioning, i.e. the CDN cannot serve content to such clients with reasonable QoS. Without server-initiated notifications, the uCDN might still use a very recent network and cost map from dCDN, and therefore redirect request to dCDN which it cannot serve. Similarly, the possibility for incremental updates would enable efficient conveyance of the aforementioned (or similar) status changes by the dCDN to the uCDN. The newest design of ALTO supports server pushed incremental updates [I-D.ietf-alto-incr-update-sse].
- o Content Availability on Hosts: A dCDN might want to express CDN capabilities in terms of certain content types (e.g. codecs/formats, or content from certain content providers). The new endpoint property for ALTO would enable a dCDN to make available such information to an upstream CDN. This would enable a uCDN to determine if a given dCDN actually has the capabilities for a given request with respect to the type of content requested.
- o Resource Availability on Hosts or Links: The capabilities on links (e.g. maximum bandwidth) or caches (e.g. average load) might be useful information for an upstream CDN for optimized downstream CDN selection. For instance, if a uCDN receives a streaming request for content with a certain bitrate, it needs to know if it is likely that a dCDN can fulfill such stringent application-level requirements (i.e. can be expected to have enough consistent bandwidth) before it redirects the request. In general, if ALTO could convey such information via new endpoint properties, it would enable more sophisticated means for downstream CDN selection with ALTO.

3. CDNI FCI ALTO Service

The ALTO protocol is based on an ALTO Information Service Framework which consists of several services, where all ALTO services are 'provided through a common transport protocol, messaging structure and encoding, and transaction model' [RFC7285]. The ALTO protocol specification [RFC7285] defines several such services, e.g. the ALTO map service.

This document defines a new ALTO Service called 'CDNI Footprint & Capabilities Advertisement Service' which conveys JSON objects of media type 'application/cdni'. This media type and JSON object

format is defined in [RFC8006] and [RFC8008]; this document specifies how to transport such JSON objects via the ALTO protocol with the ALTO 'CDNI Footprint & Capabilities Advertisement Service'.

3.1. Server Response Encoding

3.1.1. Media Type

The media type of the CDNI FCI Map is 'application/cdni'.

3.1.2. Meta Information

The 'meta' field of a FCI response MUST include 'vtag', which is an ALTO Version Tag of the retrieved FCIMapData according to [RFC7285] (Section 10.3.). It thus contains a 'resource-id' attribute, and a 'tag' is an identifier string.

3.1.3. Data Information

The data component of a CDNI FCI resource is named 'cdni-fcimap' which is a JSON object defined by [RFC8008]. This JSON object is derived from ResponseEntityBase as specified in the ALTO protocol [RFC7285] (Section 8.4.).

3.2. Protocol Errors

Protocol errors are handled as specified in the ALTO protocol [RFC7285] (Section 8.5.).

3.3. Examples

3.3.1. Basic Example

The following example shows an CDNI FCI response as in [RFC8008], however with meta-information as defined in Section 3.1.2 of this document.

```
GET /fcimap HTTP/1.1
Host: alto.example.com
Accept: application/cdni,application/alto-error+json

HTTP/1.1 200 OK
Content-Length: 439
Content-Type: application/cdni
{
  "meta" : {
    "vtag": {
      "resource-id": "my-default-fcimap",
      "tag": "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
    }
  },
  "cdni-fcimap": {
    "capabilities": [
      {
        "capability-type": "FCI.DeliveryProtocol",
        "capability-value": {
          "delivery-protocols": [
            "http/1.1",
          ]
        },
        "footprints": [
          <Footprint objects>
        ]
      }
    ]
  }
}
```

3.3.2. Incremental FCI Update Example

3.3.3. FCI Using ALTO Network Map Example

4. Security Considerations

One important security consideration is the proper authentication of advertisement information provided by a downstream CDN. The ALTO protocol provides a specification for a signature of ALTO information (see 8.2.2. of [RFC7285]). ALTO thus provides a proper means for protecting the integrity of FCI information.

More Security Considerations will be discussed in a future version of this document.

5. Acknowledgements

The authors would like to thank Kevin Ma, Daryl Malas, and Matt Caulfield for their timely reviews and invaluable comments.

Jan Seedorf is partially supported by the GreenICN project (GreenICN: Architecture and Applications of Green Information Centric Networking), a research project supported jointly by the European Commission under its 7th Framework Program (contract no. 608518) and the National Institute of Information and Communications Technology (NICT) in Japan (contract no. 167). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the GreenICN project, the European Commission, or NICT.

6. References

6.1. Normative References

- [RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, DOI 10.17487/RFC5693, October 2009, <<http://www.rfc-editor.org/info/rfc5693>>.
- [RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", RFC 6707, DOI 10.17487/RFC6707, September 2012, <<http://www.rfc-editor.org/info/rfc6707>>.
- [RFC6770] Bertrand, G., Ed., Stephan, E., Burbidge, T., Eardley, P., Ma, K., and G. Watson, "Use Cases for Content Delivery Network Interconnection", RFC 6770, DOI 10.17487/RFC6770, November 2012, <<http://www.rfc-editor.org/info/rfc6770>>.
- [RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <<http://www.rfc-editor.org/info/rfc7285>>.
- [RFC7336] Peterson, L., Davie, B., and R. van Brandenburg, Ed., "Framework for Content Distribution Network Interconnection (CDNI)", RFC 7336, DOI 10.17487/RFC7336, August 2014, <<http://www.rfc-editor.org/info/rfc7336>>.

- [RFC7337] Leung, K., Ed. and Y. Lee, Ed., "Content Distribution Network Interconnection (CDNI) Requirements", RFC 7337, DOI 10.17487/RFC7337, August 2014, <<http://www.rfc-editor.org/info/rfc7337>>.
- [RFC8006] Niven-Jenkins, B., Murray, R., Caulfield, M., and K. Ma, "Content Delivery Network Interconnection (CDNI) Metadata", RFC 8006, DOI 10.17487/RFC8006, December 2016, <<http://www.rfc-editor.org/info/rfc8006>>.
- [RFC8008] Seedorf, J., Peterson, J., Previdi, S., van Brandenburg, R., and K. Ma, "Content Delivery Network Interconnection (CDNI) Request Routing: Footprint and Capabilities Semantics", RFC 8008, DOI 10.17487/RFC8008, December 2016, <<http://www.rfc-editor.org/info/rfc8008>>.

6.2. Informative References

- [I-D.ietf-alto-incr-update-sse]
Roome, W. and Y. Yang, "ALTO Incremental Updates Using Server-Sent Events (SSE)", draft-ietf-alto-incr-update-sse-07 (work in progress), July 2017.
- [I-D.jenkins-alto-cdn-use-cases]
Niven-Jenkins, B., Watson, G., Bitar, N., Medved, J., and S. Previdi, "Use Cases for ALTO within CDNs", draft-jenkins-alto-cdn-use-cases-03 (work in progress), June 2012.
- [I-D.ma-cdni-capabilities]
Ma, K. and J. Seedorf, "CDNI Footprint & Capabilities Advertisement Interface", draft-ma-cdni-capabilities-09 (work in progress), April 2016.

Authors' Addresses

Jan Seedorf
HFT Stuttgart - Univ. of Applied Sciences
Schellingstrasse 24
Stuttgart 70174
Germany

Phone: +49-0711-8926-2801
Email: jan.seedorf@hft-stuttgart.de

Y.R. Yang
Tongji/Yale University
51 Prospect Street
New Haven, CT 06511
United States of America

Email: yry@cs.yale.edu
URI: <http://www.cs.yale.edu/~yry/>

Kevin J. Ma
Ericsson
43 Nagog Park
Acton, MA 01720
United States of America

Phone: +1-978-844-5100
Email: kevin.j.ma@ericsson.com

Jon Peterson
NeuStar
1800 Sutter St Suite 570
Concord, CA 94520
United States of America

Email: jon.peterson@neustar.biz

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 2013

M-K. Shin
H-J. Kim
ETRI
D. Chang
T. Kwon
SNU
July 1, 2012

CDNI Request Routing with SDN
draft-shin-cdni-request-routing-sdn-00

Abstract

Software-defined networking (SDN) is emerging and intensively discussed as one of the most promising technologies to provide centralized, programmable control planes for network service providers (NSPs). In this sense, SDN could be also considered as one of candidates to facilitate CDNI Request Routing. This document discusses how SDN can be used for downstream CDN selection within CDNI request routing.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on January 1, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. SDN Overview and Assumption	3
3. SDN within CDNI Request Routing	4
4. Selection of a Downstream CDN with SDN	5
5. Example of Content Request Redirection and Path Setup for Content Delivery with SDN	6
6. Advantages of using SDN	7
7. Further Considerations	7
8. Security Considerations	7
9. Acknowledgements	7
10. Informative References	7
Author's Address	9

1. Introduction

The CDNI PS and framework documents [I-D.ietf-cdni-problem-statement][I-D.ietf-cdni-framework] define the following four interfaces for CDNI; Request Routing Interface, Metadata Interface, Logging Interface, and CDNI Control Interface. As for the Request Routing - Redirection, HTTP and DNS are being discussed as one of candidate protocols. Recently, Software-defined networking (SDN) is emerging and intensively discussed as one of the most promising technologies to provide centralized, programmable control planes for network service providers (NSPs). In this sense, SDN could be also considered as one of candidates to facilitate CDNI Request Routing. This document discusses how OF/SDN technology can be integrated in CDNI request routing.

1.1. Terminology

This document draws freely on the terminology defined in [RFC3466] and [I-D.ietf-cdni-problem-statement].

We also introduce the following terms, tentatively:

CDN Frontend Server (CFS): CDN Frontend Server (CFS): It is a server which has SDN capability. A Network Service Providers (NSP) registers the address of its own CFS to DNS. In this way, a CFS can redirect "request" messages to its SDN controller.

2. SDN Overview and Assumption

SDN is a new networking technology which allows centralized, programmable control planes, so that NSPs can control and manage directly their own virtualized resources and networks without recognizing detailed hardware technologies. To achieve this, with SDN, control and data planes are separated which allows control to be directly programmable and manageable in a centralized manner and data plane to be simplified and abstracted rather than specialized hardware. Since work on SDN architecture and framework is still being discussed and is not standardized yet, in this document, it is assumed that OpenFlow is as one of architectural components for SDN framework to facilitate CDNI Request Routing, as an example, but any other existing and/or possible solutions for SDN could be also integrated.

Most modern Ethernet switches and routers contain flowtables that run at line-rate to implement firewalls, NAT, QoS, and to collect statistics. While each vendor's flowtable is different, OpenFlow's

basic idea composed an interesting common set of functions that run in many switches and routers. OpenFlow exploits this common set of functions. OpenFlow provides an open protocol to program the flowtable in different switches and routers. In an OpenFlow network, a central controller manages switches that support the concept of a flow, a stream of related packets that are processed in the same way. Every switch maintains a flow table containing a set of rules, where each rule includes a pattern that is the set of packets belonging to the flow, a priority that disambiguates overlapping rules, an expiration time, a list of actions to apply to the packets, and counters to measure the traffic. To process an incoming packet, the switch identifies the matching rule with the highest priority, updates the counters of the rule, and applies the actions. If no matching rule is found, the switch forwards the packet to the controllers and awaits further instructions.

3. SDN within CDNI Request Routing

The scope of the CDNI Request Routing Interface SHOULD contain two functionalities [I-D.ietf-cdni-framework] :

- o Request Routing Interface - Footprint and Capabilities Advertisement;
the asynchronous advertisement of footprint and capabilities by a dCDN that allows a uCDN to decide whether to redirect particular user requests to that dCDN;
- o Request Routing Interface - Redirection;
the synchronous operation of actually redirecting a user request

First of all, it is assumed that ALTO is used for Request Routing Interface - Footprint and Capabilities Advertisement. Details and examples on how a downstream CDN can advertise its footprint and other information by means of ALTO are being discussed in [I-D.seedorf-cdni-request-routing-alto]. Application Layer Traffic Optimization (ALTO) is an approach for guiding the resource provider selection process in distributed applications that can choose among several candidate resources providers to retrieve a given resource. By conveying network layer (topology) information, an ALTO server can provide important information to guide the resource provider selection process in distributed applications.

As for the Request Routing - Redirection, HTTP and DNS are being discussed as one of candidate protocols. Recently, SDN is emerging and intensively discussed as one of the most promising technologies to provide centralized, programmable control planes for network service providers (NSPs). In this sense, SDN could be also considered

as one of candidates to facilitate CDNI Request Routing as well as HTTP and DNS. This document discusses how SDN technology can be integrated in CDNI request routing.

4. Selection of a downstream CDN with SDN

SDN can help the upstream CDN provider to select a proper downstream CDN provider for a given end user request as follows. It is assumed that each downstream CDN provider hosts SDN controller.

An example of operation is as follows :

0) dCDN advertises information relevant to its delivery capabilities (e.g. content availability, geographic footprint, etc.) using ALTO extension (e.g., I2AEX) provisioning prior to any content requests being redirected.

1) A content request from a user agent arrives in the CFS of uCDN.

2) The CFS at uCDN relays the message to the its SDN controller by a "Packet-In" message.

3) ALTO client at the OF controller requests the best dCDN information to ALTO server (ALTO cost map and/or other information may be used).

4) ALTO server responses and then OF controller in uCDN knows which is the best dCDN.

5) The SDN controller sends a query to the SDN controller of the best dCDN

6) (uCDN redirects the request to the best dCDN).

5. Example of Content Request Redirection and Path Setup for Content Delivery with SDN

SDN can help the upstream CDN provider to redirect a content request message to a downstream CDN provider for a given end user request as follows. It is assumed that the upstream and the downstream CDN providers have SDN controllers. And OF/SDN sets up the path for the content delivery.

An example of operation is as follows :

0) Content distribution metadata is pre-positioned between CDNs prior

to any content requests being redirected; that is, a controller in uCDN knows its own surrogates in other CDNs and information relevant to its delivery capabilities (e.g. geo-blocking information, availability windows, desired distribution policy, etc.)

1) An end user issues an HTTP GET message to get content. By contacting DNS, this message is forwarded to the CDN frontend server (CFS) of uCDN which has SDN capability.

2) The CFS of uCDN relays this message to its own SDN controller by "Packet-In" message in SDN protocol.

3) The controller of uCDN checks content distribution metadata, and sends a query to the controller of dCDN whether it can deliver the content. In the query, the source address of the request packets (i.e. the host address of the user agent) is included. Optionally, a QoS requirement for the content delivery may be specified in the query as well. And there can be multiple candidate dCDNs for a given user request.

4) If the SDN controller of dCDN decides to provide content, it checks the location of the content object and network traffic status. And it assigns an IP address for the content delivery. The assigned IP address will be used as the content identifier, which is the source address of the data packets. After that, it sends a reply to the controller of uCDN with these data and sets up the path for content delivery from the surrogate in dCDN to the end user.

5) The SDN controller in uCDN informs the end user of the URL of the surrogate in dCDN by sending HTTP Redirection.

6) The end user sends HTTP GET message to the surrogate in dCDN.

6. Advantages of using SDN

The following reasons make SDN a suitable candidate protocol for downstream CDN selection as part of CDNI request routing:

- o Synchronous CDNI operations
- o Integrated with SDN framework and architecture (e.g., OpenFlow)
- o Traffic isolated with desired QoS/QoE, security, etc.
- o More extensible (suitable for i2aex proposal)

- o More centralized, programmable (e.g., using SDN Apps for CDNI)
- o Mobility support

7. Further Considerations

The following further issues should be also discussed on SDN architecture and framework for downstream CDN selection as part of CDNI request routing:

- o ALTO extension (i2aex)
- o Northbound interfaces of SDN controllers
- o Multi-controllers
- o East-west bound interfaces of SDN controllers

8. Security Considerations

TBD

9. Acknowledgements

TBD

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

10.2. Informative References

- [I-D.ietf-cdni-problem-statement] Niven-Jenkins, B., Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", draft-ietf-cdni-problem-statement-04 (work in progress), March 2012.

- [I-D.ietf-cdni-requirements] Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", draft-ietf-cdni-requirements-02 (work in progress),

December 2011.

[I-D.ietf-cdni-framework] Peterson, L. and B. Davie, "Framework for CDN Interconnection", draft-ietf-cdni-framework-00 (work in progress), April 2012.

[I-D.seedorf-cdni-request-routing-alto] Seedorf, J., "CDNI Request Routing with ALTO", draft-seedorf-cdni-request-routing-alto-01 (work in progress), March 2012.

[RFC3466] Day, M., Cain, B., Tomlinson, G., and P. Rzewski, "A Model for Content Internetworking (CDI)", RFC 3466, February 2003.

[RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, October 2009.

[b-OpenFlow] OpenFlow Switch Specification 1.3,
<http://www.opennetworking.org/>.

Authors' Addresses

Myung-Ki Shin
ETRI
161 Gajeong-dong Yuseng-gu
Daejeon, 305-700
Korea

Phone: +82 42 860 4847
Email: mkshin@etri.re.kr

Hyoung-Jun Kim
ETRI
161 Gajeong-dong Yuseng-gu
Daejeon, 305-700
Korea

Phone: +82 42 860 48476576
Email: khj@etri.re.kr

Dukhyun Chang
Seoul National University
Seoul, Korea

Email: dhchang@mmlab.snu.ac.kr

Ted Taekyoung Kwon
Seoul National University
Seoul, Korea

Email: tkkwon98@gmail.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: October 2013

M-K. Shin
S. Lee
ETRI
D. Chang
T. Kwon
SNU
February 15, 2013

CDNI Request Routing with SDN
draft-shin-cdni-request-routing-sdn-01

Abstract

Software-defined networking (SDN) is emerging and intensively discussed as one of the most promising technologies to provide centralized, programmable control planes for network service providers (NSPs). In this sense, SDN could be also considered as one of candidates to facilitate CDNI Request Routing. This document discusses how SDN can be used for downstream CDN selection within CDNI request routing.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on January 1, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. SDN Overview and Assumption	3
3. SDN within CDNI Request Routing	4
4. Selection of a Downstream CDN with SDN	5
5. Example of Content Request Redirection and Path Setup for Content Delivery with SDN	6
6. Advantages of using SDN	7
7. Further Considerations	7
8. Security Considerations	7
9. Acknowledgements	7
10. Informative References	7
Author's Address	9

1. Introduction

The CDNI PS and framework documents [RFC6707][I-D.ietf-cdni-framework] define the following four interfaces for CDNI; Request Routing Interface, Metadata Interface, Logging Interface, and CDNI Control Interface. As for the Request Routing - Redirection, HTTP and DNS are being discussed as one of candidate protocols. Recently, Software-defined networking (SDN) is emerging and intensively discussed as one of the most promising technologies to provide centralized, programmable control planes for network service providers (NSPs). In this sense, SDN could be also considered as one of candidates to facilitate CDNI Request Routing. This document discusses how OF/SDN technology can be integrated in CDNI request routing.

1.1. Terminology

This document draws freely on the terminology defined in [RFC3466] and [RFC6706].

We also introduce the following terms, tentatively:

CDN Frontend Server (CFS): CDN Frontend Server (CFS): It is a server which has SDN capability. A Network Service Providers (NSP) registers the address of its own CFS to DNS. In this way, a CFS can redirect "request" messages to its SDN controller.

2. SDN Overview and Assumption

SDN is a new networking technology which allows centralized, programmable control planes, so that NSPs can control and manage directly their own virtualized resources and networks without recognizing detailed hardware technologies. To achieve this, with SDN, control and data planes are separated which allows control to be directly programmable and manageable in a centralized manner and data plane to be simplified and abstracted rather than specialized hardware. Since work on SDN architecture and framework is still being discussed and is not standardized yet, in this document, it is assumed that OpenFlow is as one of architectural components for SDN framework to facilitate CDNI Request Routing, as an example, but any other existing and/or possible solutions for SDN, such as I2RS and I2AEX could be also integrated without any big modifications.

Most modern Ethernet switches and routers contain flowtables that run at line-rate to implement firewalls, NAT, QoS, and to collect statistics. While each vendor's flowtable is different, OpenFlow's

basic idea composed an interesting common set of functions that run in many switches and routers. OpenFlow exploits this common set of functions. OpenFlow provides an open protocol to program the flowtable in different switches and routers. In an OpenFlow network, a central controller manages switches that support the concept of a flow, a stream of related packets that are processed in the same way. Every switch maintains a flow table containing a set of rules, where each rule includes a pattern that is the set of packets belonging to the flow, a priority that disambiguates overlapping rules, an expiration time, a list of actions to apply to the packets, and counters to measure the traffic. To process an incoming packet, the switch identifies the matching rule with the highest priority, updates the counters of the rule, and applies the actions. If no matching rule is found, the switch forwards the packet to the controllers and awaits further instructions.

3. SDN within CDNI Request Routing

The scope of the CDNI Request Routing Interface SHOULD contain two functionalities [I-D.ietf-cdni-framework] :

- o Request Routing Interface - Footprint and Capabilities Advertisement;
the asynchronous advertisement of footprint and capabilities by a dCDN that allows a uCDN to decide whether to redirect particular user requests to that dCDN;
- o Request Routing Interface - Redirection;
the synchronous operation of actually redirecting a user request

First of all, it is assumed that ALTO is used for Request Routing Interface - Footprint and Capabilities Advertisement. Details and examples on how a downstream CDN can advertise its footprint and other information by means of ALTO are being discussed in [I-D.seedorf-cdni-request-routing-alto]. Application Layer Traffic Optimization (ALTO) is an approach for guiding the resource provider selection process in distributed applications that can choose among several candidate resources providers to retrieve a given resource. By conveying network layer (topology) information, an ALTO server can provide important information to guide the resource provider selection process in distributed applications.

As for the Request Routing - Redirection, HTTP and DNS are being discussed as one of candidate protocols. Recently, SDN is emerging and intensively discussed as one of the most promising technologies to provide centralized, programmable control planes for network service providers (NSPs). In this sense, SDN could be also considered

as one of candidates to facilitate CDNI Request Routing as well as HTTP and DNS. This document discusses how SDN technology can be integrated in CDNI request routing.

4. Selection of a downstream CDN with SDN

SDN can help the upstream CDN provider to select a proper downstream CDN provider for a given end user request as follows. It is assumed that each downstream CDN provider hosts SDN controller.

An example of operation is as follows :

0) dCDN advertises information relevant to its delivery capabilities (e.g. content availability, geographic footprint, etc.) using ALTO extension (e.g., I2AEX) provisioning prior to any content requests being redirected.

1) A content request from a user agent arrives in the CFS of uCDN.

2) The CFS at uCDN relays the message to the its SDN controller by a "Packet-In" message.

3) ALTO client at the OF controller requests the best dCDN information to ALTO server (ALTO cost map and/or other information may be used).

4) ALTO server responses and then OF controller in uCDN knows which is the best dCDN.

5) The SDN controller sends a query to the SDN controller of the best dCDN

6) (uCDN redirects the request to the best dCDN).

5. Example of Content Request Redirection and Path Setup for Content Delivery with SDN

SDN can help the upstream CDN provider to redirect a content request message to a downstream CDN provider for a given end user request as follows. It is assumed that the upstream and the downstream CDN providers have SDN controllers. And OF/SDN sets up the path for the content delivery.

An example of operation is as follows :

0) Content distribution metadata is pre-positioned between CDNs prior

to any content requests being redirected; that is, a controller in uCDN knows its own surrogates in other CDNs and information relevant to its delivery capabilities (e.g. geo-blocking information, availability windows, desired distribution policy, etc.)

1) An end user issues an HTTP GET message to get content. By contacting DNS, this message is forwarded to the CDN frontend server (CFS) of uCDN which has SDN capability.

2) The CFS of uCDN relays this message to its own SDN controller by "Packet-In" message in SDN protocol.

3) The controller of uCDN checks content distribution metadata, and sends a query to the controller of dCDN whether it can deliver the content. In the query, the source address of the request packets (i.e. the host address of the user agent) is included. Optionally, a QoS requirement for the content delivery may be specified in the query as well. And there can be multiple candidate dCDNs for a given user request.

4) If the SDN controller of dCDN decides to provide content, it checks the location of the content object and network traffic status. And it assigns an IP address for the content delivery. The assigned IP address will be used as the content identifier, which is the source address of the data packets. After that, it sends a reply to the controller of uCDN with these data and sets up the path for content delivery from the surrogate in dCDN to the end user.

5) The SDN controller in uCDN informs the end user of the URL of the surrogate in dCDN by sending HTTP Redirection.

6) The end user sends HTTP GET message to the surrogate in dCDN.

6. Advantages of using SDN

The following reasons make SDN a suitable candidate protocol for downstream CDN selection as part of CDNI request routing:

- o Synchronous CDNI operations
- o Integrated with SDN framework and architecture (e.g., OpenFlow)
- o Traffic isolated with desired QoS/QoE, security, etc.
- o More extensible (suitable for i2aex proposal)

- o More centralized, programmable (e.g., using SDN Apps for CDNI)
- o Mobility support

7. Further Considerations

The following further issues should be also discussed on SDN architecture and framework for downstream CDN selection as part of CDNI request routing:

- o ALTO extension (i2aex)
- o Northbound interfaces of SDN controllers
- o Multi-controllers
- o East-west bound interfaces of SDN controllers

8. Security Considerations

TBD

9. Acknowledgements

TBD

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

10.2. Informative References

- [RFC6707] Niven-Jenkins, B., Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", RFC 6706, September 2012.
- [I-D.ietf-cdni-requirements] Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", draft-ietf-cdni-requirements-02 (work in progress), December 2011.

- [I-D.ietf-cdni-framework] Peterson, L. and B. Davie, "Framework for CDN Interconnection", draft-ietf-cdni-framework-00 (work in progress), April 2012.
- [I-D.seedorf-cdni-request-routing-alto] Seedorf, J., "CDNI Request Routing with ALTO", draft-seedorf-cdni-request-routing-alto-01 (work in progress), March 2012.
- [RFC3466] Day, M., Cain, B., Tomlinson, G., and P. Rzewski, "A Model for Content Internetworking (CDI)", RFC 3466, February 2003.
- [RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, October 2009.
- [b-OpenFlow] OpenFlow Switch Specification 1.3,
<http://www.opennetworking.org/>.

Authors' Addresses

Myung-Ki Shin
ETRI
161 Gajeong-dong Yuseng-gu
Daejeon, 305-700
Korea

Phone: +82 42 860 4847
Email: mkshin@etri.re.kr

Seungik Lee
ETRI
161 Gajeong-dong Yuseng-gu
Daejeon, 305-700
Korea

Phone: +82 42 860 1483
Email: seungiklee@etri.re.kr

Dukhyun Chang
Seoul National University
Seoul, Korea

Email: dhchang@mmlab.snu.ac.kr

Ted Taekyoung Kwon
Seoul National University
Seoul, Korea

Email: tkkwon98@gmail.com

CDNI
Internet-Draft
Intended status: Standards Track
Expires: January 17, 2013

H. Song
Huawei
Y. Zhang
China Mobile
July 16, 2012

A SLR (Service Level Requirements) based footprint for CDNI
draft-song-cdni-slr-based-footprint-01

Abstract

Footprint advertisement is a very important step for CDN interconnection and generates a lot of discussion. Actually, each CDN can serve the whole world if its surrogates are publicly reachable by IP addresses. But if a CDN does that, it can not satisfy the requirements from the applications. So CDNs deliver contents for applications, and the basic requirements should be from the applications, but there is rare discussion on service level requirements based footprint. This document is used to generate the discussion on this aspect.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 17, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Terminology	3
2. Why SLR Based Footprint	3
3. What are the Parameters for SLR	4
3.1. Average Response Time	4
3.2. Throughput	4
3.3. Startup Delay	4
3.4. Average downloading rate	4
3.5. Hit Ratio	5
3.6. Capability	5
3.7. Up-time	5
3.8. Discussion	5
4. Dynamic Mapping for Footprint	5
5. Message Flows	6
6. Security Considerations	6
7. IANA Considerations	6
8. Normative References	6
Authors' Addresses	7

1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Why SLR Based Footprint

Each CDN's footprint can be worldwide, if its surrogates' IP addresses are publicly reachable. However, not every CDN can serve the applications for worldwide distribution because it can not satisfy the service level required by those applications. So what an application basically needs is a CDN to satisfy its service level, and distribute the contents to certain areas. If a CDN or together with its downstream CDNs, cannot meet the SLR (service level requirements) in an area from an application, then we can say this upstream CDN is not competent for this content distribution task. This document specifies how the parameters of SLR impact a CDN's footprint. There is other draft [I-D.he-cdni-cap-info-advertising] mentioned capability advertisement, please note that capability advertisement is also very important and footprint is impacted by capability of a CDN. While each CDN serve many tasks concurrently, the dynamic resources that it can allocate is also variable at different time.

The physical deployment area of a CDN might be small, but it can have larger footprint area where it can satisfy an application's SLR. The footprint area might be even larger than a CDN that has larger physical deployment area. Choosing SLR as the basis for footprint can avoid some CDN magnifying its service level and service area on purpose, and also make some other "small" but powerful CDN be treated with justness.

We think that applications should participate in the CDN interconnection process implicitly, i.e. its requirements for service level should be transmitted between upstream and downstream CDNs (message protection is required due to the privacy). A downstream CDN should notify its capability information to its upstream CDN as well when notifying its footprint that satisfies certain SLR, which will allow a upstream CDN to choose multiple downstream CDN to fulfill a task even in a same area.

From the application's perspective, a file downloading application may not care about when the user receive the first bit, but more care about the average downloading rate. While a streaming application may have different opinion. So for a same CDN, it can serve the file downloading application well with one wide footprint and serve

streaming application well with another smaller footprint.

In general, service level is the main driver for the definition of footprint, and applications do not care about the locations where a CDN's surrogates are deployed while it can satisfy its service requirements. And topologically, ALTO [I-D.ietf-alto-protocol] is used for the appropriate surrogate selection after the footprints are defined. And ALTO network map information can also be used for the footprint description to upstream CDN .

3. What are the Parameters for SLR

The general principal for SLR is fast, scalable, secure and reliable. But it needs detailed measurement metrics for it. Here we put the capability requirements as one parameter for SLR, as one upstream CDN can choose multiple downstream CDNs to satisfy an customer application's requirements. This section lists the possible parameters for SLR. However, this document is not going to define the specifics for the measurement methods.

3.1. Average Response Time

This value is to reflect the average response time in normal network condition. This value impacts the footprint a lot.

3.2. Throughput

This parameter will also impact the footprint. If a CDN's available throughput is very big then it can serve more than its deployment area.

3.3. Startup Delay

This parameter is a very important metric for the streaming media delivery. As a TCP connection throughput close to MTU/RTT. Long distance transport maybe mean smaller MTU and longer RTT, as well as more packet lost rate, which will result in a low rate data transport, and in consequence long startup delay.

3.4. Average downloading rate

Application usually needs the CDN to guarantee a certain downloading rate for a certain service. More discussion is needed on this parameter and how it impacts footprint.

3.5. Hit Ratio

This parameter is about the content availability. High hit ratio means more local service and low burden on original servers. This parameter is more related to the CDN's optimization policies than to the footprint.

3.6. Capability

Please refer other documents for the CDN capability advertisement in CDNI WG.

3.7. Up-time

Uptime is a measure of the time a machine has been up without any downtime. For a CDN system, it usually needs to guarantee a 100% up-time for system (not for each host).

3.8. Discussion

Not all parameters required for a certain service level are listed. More discussion is needed. Some parameters might impact a CDN's footprint, and some will not. Should all of them or just a portion that affect the footprint be conveyed in the same way among CDNs?

4. Dynamic Mapping for Footprint

Each CDN participate in the CDN interconnection network should maintain a map between the SLR parameters and its footprint. There are choices to exchange the map information.

(1) An application's SLR is directly sent from upstream CDN to the downstream CDN. So that downstream CDN can report its footprint to upstream CDN accordingly. The downstream CDN should guarantee it satisfies the SLR for users in its reported footprint. Although this method requires message exchange for each application, but it is simple to implement.

(2) Each CDN report its mapping between SLR parameters and footprint to upstream CDN. And the upstream CDN will make the final decision on the downstream CDN's footprint according to a specific application's SLR. This method reduces the message exchanges but the map itself might be very complicated, due to various combination of these parameter values exist.

5. Message Flows

TBD.

6. Security Considerations

These security issues are open for discussion:

(1) Applications might take its service level requirements as a confidential? Although it can be a confidential to users, but it can be protected without leaking to any third party that is not involved in the CDN interconnection?

(2) CDNs might take its footprint according to SLR as confidential?

(3) Footprint cheating. A CDN may cheat with its footprint. If the behavior is discovered, the application cannot get the service level in that announced footprint, punishment policies should be applied to the CDN provider.

7. IANA Considerations

There is no IANA consideration for this document.

8. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[I-D.he-cdni-cap-info-advertising]
He, X., Dawkins, S., Chen, G., Zhang, Y., and W. Ni,
"Capability Information Advertising for CDN
Interconnection", draft-he-cdni-cap-info-advertising-01
(work in progress), March 2012.

[I-D.seedorf-cdni-request-routing-alto]
Seedorf, J., "CDNI Request Routing with ALTO",
draft-seedorf-cdni-request-routing-alto-01 (work in
progress), March 2012.

[I-D.ietf-alto-protocol]
Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol",
draft-ietf-alto-protocol-12 (work in progress), July 2012.

Authors' Addresses

Haibin Song
Huawei

Email: haibin.song@huawei.com

Yunfei Zhang
China Mobile

Email: zhangyunfei@chinamobile.com

CDNI
Internet-Draft
Intended status: Standards Track
Expires: March 7, 2013

H. Song
Huawei
Y. Zhang
China Mobile
Y. Sun
ICT/CAS
Sep 3, 2012

A SLR (Service Level Requirements) based footprint for CDNI
draft-song-cdni-slr-based-footprint-02

Abstract

Footprint advertisement is a very important step for CDN interconnection and generates a lot of discussion. Actually, each CDN can serve the whole world if its surrogates are publicly reachable by IP addresses. But if a CDN does that, it can not satisfy the requirements from the applications. So CDNs deliver contents for applications, and the basic requirements should be from the applications. One CDN can serve different applications well with different footprint. But there is rare discussion on service level requirements based footprint. This document is used to generate the discussion on this aspect.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 7, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal

Provisions Relating to IETF Documents
(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Terminology	3
2. Why SLR Based Footprint	3
3. What are the Parameters for SLR	4
3.1. Average Response Time	4
3.2. Throughput	4
3.3. Startup Delay	4
3.4. Average downloading rate	5
3.5. Hit Ratio	5
3.6. Capability	5
3.7. Up-time	5
3.8. Discussion	5
4. Dynamic Mapping for Footprint	5
5. Message Flows	6
6. Security Considerations	6
7. IANA Considerations	6
8. Normative References	6
Authors' Addresses	7

1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Why SLR Based Footprint

Each CDN's footprint can be worldwide, if its surrogates' IP addresses are publicly reachable. However, not every CDN can serve the applications for worldwide distribution because it can not satisfy the server level required by those applications. So what an application basically needs is a CDN to satisfy its service level, and distribute the contents to certain areas. If a CDN or together with its downstream CDNs, cannot meet the SLR (service level requirements) in an area from an application, then we can say this upstream CDN is not competent for this content distribution task. This document specifies how the parameters of SLR impact a CDN's footprint. There is other draft [I-D.he-cdni-cap-info-advertising] mentioned capability advertisement, please note that capability advertisement is also very important and footprint is impacted by capability of a CDN. We consider capability as one service requirement factor from applications. While each CDN serve many tasks concurrently, the dynamic resources that it can allocate is also variable at different time.

The physical deployment area of a CDN might be small, but it can have larger footprint area where it can satisfy an application's SLR. The footprint area might be even larger than a CDN that has larger physical deployment area. Choosing SLR as the basis for footprint can avoid some CDN magnifying its service level and service area on purpose, and also make some other "small" but powerful CDN be treated with justness.

We think that applications should participate in the CDN interconnection process implicitly, i.e. its requirements for service level should be transmitted between upstream and downstream CDNs (message protection is required due to the privacy). A downstream CDN should notify its capability information to its upstream CDN as well when notifying its footprint that satisfies certain SLR, which will allow a upstream CDN to choose multiple downstream CDN to fullfill a task even in a same area.

From the application's perspective, a file downloading application may not care about when the user receive the first bit, but more care about the average downloading rate. While a streaming application may have a different opinion. So for a same CDN, it can serve the

file downloading application well with one wider footprint and serve streaming application well with another smaller footprint.

In general, service level is the main driver for the definition of footprint, and applications do not care about the locations where a CDN's surrogates are deployed while it can satisfy its service requirements. And topologically, ALTO [I-D.ietf-alto-protocol] is used for the appropriate surrogate selection after the footprints are defined. And ALTO network map information can also be used for the footprint description to upstream CDN .

3. What are the Parameters for SLR

The general principal for service level requirements is fast, scalable, secure and reliable. But it needs detailed measurement metrics for it. Here we put the capability requirements as one parameter for SLR, as one upstream CDN can choose multiple downstream CDNs to satisfy an customer application's requirements. It does not matter that much if with one footprint, one downstream CDN can satisfy the performance requirements but not capability requirements. This section lists the possible parameters for SLR.

We consider the parameters that are not high dynamic. Those parameters that are dynamic at a very brief time frame, but statistically rather static at a reasonable time frame (like one month) can be considered for footprint determination.

However, this document is not going to define the specifics for the measurement methods.

3.1. Average Response Time

This value is to reflect the average response time in normal network condition. This value impacts the footprint a lot.

3.2. Throughput

This parameter will also impact the footprint. If a CDN's available throughput is very big then it can serve more than its deployment area.

3.3. Startup Delay

This parameter is a very important metric for the streaming media delivery. As a TCP connection throughput close to MTU/RTT. Long distance transport maybe mean smaller MTU and longer RTT, as well as more packet lost rate, which will result in a low rate data

transport, and in consequence long startup delay.

3.4. Average downloading rate

Application usually needs the CDN to guarantee a certain downloading rate for a certain service. More discussion is needed on this parameter and how it impacts footprint.

3.5. Hit Ratio

This parameter is about the content availability. High hit ratio means more local service and low burden on original servers. This parameter is more related to the CDN's optimization policies than to the footprint.

3.6. Capability

For the capability, we consider the processing power of the CDN and its features that can support certain kinds of applications.

3.7. Up-time

Uptime is a measure of the time a machine has been up without any downtime. For a CDN system, it usually needs to guarantee a 100% up-time for system (not for each host).

3.8. Discussion

Not all parameters required for a certain service level are listed. More discussion is needed. Some parameters might impact a CDN's footprint, and some will not. Should all of them or just a portion that affect the footprint be conveyed in the same way among CDNs?

4. Dynamic Mapping for Footprint

Each CDN participate in the CDN interconnection network should maintain a map between the SLR parameters and its footprint. There are choices to exchange the map information.

(1) An application's SLR is directly sent from upstream CDN to the downstream CDN. So that downstream CDN can report its footprint to upstream CDN accordingly. The downstream CDN should guarantee it satisfies the SLR for users in its reported footprint. Although this method requires message exchange for each application, but it is simple to implement.

(2) Each CDN report its mapping between SLR parameters and footprint

to upstream CDN. And the upstream CDN will make the final decision on the downstream CDN's footprint according to a specific application's SLR. This method reduces the message exchanges but the map itself might be very complicated, due to various combination of these parameter values exist.

5. Message Flows

TBD.

6. Security Considerations

These security issues are open for discussion:

(1) Applications might take its service level requirements as a confidential? Although it can be a confidential to users, but it can be protected without leaking to any third party that is not involved in the CDN interconnection?

(2) CDNs might take its footprint according to SLR as confidential?

(3) Footprint cheating. A CDN may cheat with its footprint. If the behavior is discovered, the application cannot get the service level in that announced footprint, punishment policies should be applied to the CDN provider.

7. IANA Considerations

There is no IANA consideration for this document.

8. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[I-D.he-cdni-cap-info-advertising]
He, X., Dawkins, S., Chen, G., Zhang, Y., and W. Ni,
"Capability Information Advertising for CDN
Interconnection", draft-he-cdni-cap-info-advertising-01
(work in progress), March 2012.

[I-D.seedorf-cdni-request-routing-alto]
Seedorf, J., "CDNI Request Routing with ALTO",
draft-seedorf-cdni-request-routing-alto-02 (work in

progress), July 2012.

[I-D.ietf-alto-protocol]

Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol",
draft-ietf-alto-protocol-12 (work in progress), July 2012.

Authors' Addresses

Haibin Song
Huawei

Email: haibin.song@huawei.com

Yunfei Zhang
China Mobile

Email: zhangyunfei@chinamobile.com

Yi Sun
ICT/CAS

Email: sunyi@ict.ac.cn

CDNI
Internet-Draft
Intended status: Informational
Expires: January 17, 2013

J. Seedorf
NEC
J. Peterson
Neustar
S. Previdi
Cisco
July 16, 2012

CDNI Request Routing: Footprint and Capabilities Semantics
draft-spp-cdni-rr-foot-cap-semantics-01

Abstract

This document tries to capture the semantics of the "Footprint and Capabilities Advertisement" part of the CDNI Request Routing interface, i.e. the desired meaning and what "Footprint and Capabilities Advertisement" is expected to offer within CDNI. The discussion in this document has the goal to facilitate the choosing of one or more suitable protocols for "Footprint and Capabilities Advertisement" within CDNI Request Routing.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 17, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Apparent Understanding of CDNI Footprint and Capabilities Advertisement	4
2.1. Description of footprint and capabilities advertisement in existing CDNI documents	4
2.2. Summary of understanding of footprint and capabilities advertisement in existing CDNI documents	6
3. Design Decisions for Footprint and Capabilities	7
3.1. Advertising Limited Coverage	7
3.2. Capabilities and Dynamic Data	8
3.3. Advertisement versus Queries	9
3.4. Avoiding or Handling 'cheating' Downstream CDNs	9
3.5. Focus on Main Use Cases may Simplify Things	10
4. Towards Semantics for Footprint Advertisement	11
5. Towards Semantics for Capabilities Advertisement	12
6. Open Issues and Questions	14
7. Security Considerations	15
8. Conclusion	16
9. References	17
9.1. Normative References	17
9.2. Informative References	17
Appendix A. Acknowledgment	18
Authors' Addresses	19

1. Introduction

The CDNI working group is working on a set of protocols to enable the interconnection of multiple CDNs to a CDN federation. This CDN-federation should serve multiple purposes, as discussed in [I-D.ietf-cdni-use-cases], for instance, to extend the reach of a given CDN to areas in the network which are not covered by this particular CDN.

The goal of this document is to achieve a clear understanding in the CDNI WG about the semantics associated with the CDNI request routing interface, in particular regarding the "footprint and capabilities advertisement" of a downstream CDN. To narrow down undecided aspects of these semantics, this document first tries to capture the common understanding of what the "footprint and capabilities advertisement" should offer and accomplish, i.e. what seems to be agreed. Then, the document will discuss open questions. It is the goal of this document to capture the outcome of discussions and answers to open questions in future versions of this draft. In particular, this document summarizes the progress of the recently formed CDNI design team on "footprint and capabilities advertisement".

General assumptions in this document:

- o The CDNs participating in the CDN federation have already performed a boot strap process, i.e., they have connected to each other, either directly or indirectly, and can exchange information amongst each other.
- o The uCDN has received footprint and/or capability advertisements from a set of dCDNs. Footprint advertisement and capability advertisement need not use the same underlying protocol.
- o The upstream CDN (uCDN) receives the initial request-routing request from the endpoint requesting the resource.

This document is organized as follows. We first recap the descriptions regarding "footprint and capabilities advertisement" in existing documents and try to distill the apparent common understanding of the terms "footprint" and "capabilities" in the CDNI request routing context. We then separately discuss the semantics of the footprint advertisement mechanism, and the capability advertisement mechanism. Finally, we list open issues and questions to be discussed in the CDNI WG.

Comments and discussions about this memo should be directed to the CDNI WG: cdni@ietf.org.

2. Apparent Understanding of CDNI Footprint and Capabilities Advertisement

In the following, we will summarize the descriptions of the CDNI "footprint and capabilities advertisement" as part of the "request routing" interface in existing documents. We will then carve out the apparent common understanding of what this interface is intended to offer and accomplish.

2.1. Description of footprint and capabilities advertisement in existing CDNI documents

The CDNI problem statement draft [I-D.ietf-cdni-problem-statement] describes footprint and capabilities advertisement as: "enabling an upstream CDN to determine if a downstream CDN is able (and willing) to accept the delegated content request". In addition, the draft says "the CDNI Request Routing interface is also expected to enable a downstream CDN to provide to the upstream CDN (static or dynamic) information (e.g. resources, footprint, load) to facilitate selection of the downstream CDN by the upstream CDN request routing system when processing subsequent content requests from User Agents". It thus considers "resources" and "load" as capabilities to be advertised by the downstream CDN.

The CDNI use cases draft [I-D.ietf-cdni-use-cases] describes capabilities as "... supported range of devices and User Agents or the supported range of delivery technologies". Examples for such capabilities given are specific delivery protocols, technology migration, and meeting a certain QoS.

The CDNI requirements draft [I-D.ietf-cdni-requirements] lists several requirements relevant for the "footprint and capabilities advertisement" part of the CDNI request routing interface. In summary, the following requirements for the CDNI Request Routing Interface and general requirements are relevant for the understanding of the semantics of the "footprint and capabilities advertisement":

- o GEN-4 [HIGH], "The CDNI solution shall not require intra-CDN information to be exposed to other CDNs for effective and efficient delivery of the content. Examples of intra-CDN information include surrogate topology, surrogate status, cached content, etc."
- o GEN-9 [MED], "The CDNI solution should support cascaded CDN redirection (CDN1 redirects to CDN2 that redirects to CDN3) to an arbitrary number of levels beyond the first level."

- o GEN-10 [MED], "The CDNI solution should support an arbitrary topology of interconnected CDNs (i.e. the CDN topology cannot be restricted to a tree, a loop-free topology, etc.)."
- o GEN-11 [HIGH], "The CDNI solution shall prevent looping of any CDNI information exchange."
- o REQ-1 [HIGH], allowing the downstream CDN "to communicate to the Upstream CDN coarse information about the Downstream CDN ability and/or willingness to handle requests from the Upstream CDN. For example, this could potentially include a binary signal ("Downstream CDN ready/not-ready to take additional requests from Upstream CDN") to be used in case of excessive load or failure condition in the Downstream CDN."
- o REQ-2 [MED], allowing the downstream CDN to communicate capabilities such as supported content types and delivery protocols, a set of metrics/attributes (e.g. Streaming bandwidth, storage resources, distribution and delivery priority), a set of affinities (e.g. Preferences, indication of distribution/delivery fees), information to facilitate request redirection, as well as footprint information (e.g. "layer-3 coverage").
- o REQ-3 [MED], "In the case of cascaded redirection, the CDNI Request-Routing interface shall allow the Downstream CDN to also include in the information communicated to the Upstream CDN, information on the capabilities, resources and affinities of CDNs to which the Downstream CDN may (in turn) redirect requests received by the Upstream CDN. In that case, the CDNI Request-Routing interface shall prevent looping of such information exchange."
- o REQ-4 [LOW], allowing the downstream CDN to communicate "aggregate information on CDNI administrative limits and policy" (e.g. the maximum number of requests redirected by the Upstream CDN to be served simultaneously by the Downstream CDN or maximum aggregate volume of content (e.g. in Terabytes) to be delivered by the Downstream CDN over a time period).
- o REQ-11 [LOW], "The CDNI Request-Routing protocol may support a mechanism allowing an Upstream CDN to avoid redirecting a request to a Downstream CDN if that is likely to result in the total redirection time exceeding some limit."

Note that in REQ-2 [MED] "Layer-3 coverage" is given as an example of what "footprint" information might convey in the CDNI requirements draft [I-D.ietf-cdni-requirements]. Also, note that REQ-3 [MED] addresses cascaded (transitive) downstream CDNs. In such a case, a

downstream CDN needs to include (in its advertisement information that it conveys to an upstream CDN) also information the footprint and capabilities of any further transitive downstream CDN. Such information may be included implicitly (i.e. the cascaded dCDN is oblivious to the uCDN), or explicitly (i.e. the cascaded dCDN of the fact that there is a cascaded dCDN is visible to the uCDN). In any case, a logic is needed to process incoming footprint information from a cascaded dCDN and decide if/how it is to be re-advertised/aggregated when advertising footprint to an upstream CDN.

The CDNI framework draft [I-D.davie-cdni-framework] describes a "footprint" as in [I-D.previdi-cdni-footprint-advertisement], consisting of two parts: 1) "a class of end user requests (represented, for example, by a set of IP prefixes, or a geographic region) that the dCDN is willing and able to serve directly, without use of another dCDN", and 2) "the connectivity of the dCDN to other CDNs that may be able to serve content to users on behalf of dCDN". The term "connectivity" has recently been replaced with "reachability" in [I-D.previdi-cdni-footprint-advertisement]. Further, examples for capabilities are "the ability to handle certain types of content (e.g. specific streaming formats) or quality of service (QoS)."

2.2. Summary of understanding of footprint and capabilities advertisement in existing CDNI documents

In summary, neither the term "footprint" nor "capabilities" are clearly defined. Also, a very broad range of potential capabilities is listed in the existing documents.

3. Design Decisions for Footprint and Capabilities

A large part of the difficulty lies in understanding what we mean when try to define footprint to terms of "coverage" or "reachability." While the operators of CDNs pick strategic locations to situate caches, a cache with a public IPv4 address is reachable by any endpoint on the Internet unless some policy enforcement precludes the use of the cache.

Some CDNs aspire to cover the entire world, henceforth global CDNs; which we will henceforth call global CDNs. The footprint advertised by such a CDN in the CDNi environment would, from a coverage or reachability perspective, presumably cover all prefixes. More interesting for CDNi use cases, however, are CDNs that claim a more limited coverage, but seek to federate with other CDNs in order to create a single CDN fabric which shares resources fairly.

The key to understanding the semantics of footprint and capability advertisement lies in understand why a dCDN would advertise a limited coverage area, and how a uCDN would use such advertisements to decide among one of several dCDNs.

3.1. Advertising Limited Coverage

The basic use case that would motivate a dCDN to advertise a limited coverage is that the CDN was built to cover only a particular portion of the Internet. For example, an ISP could purpose-build a CDN to serve only their own customers by situating caches in close topological proximity to high concentrations of their subscribers. The ISP knows the prefixes it has allocated to end users and thus can easily construct a list of prefixes that its caches were positioned to serve.

When such a purpose-built CDN joined a federation, however, and advertises its footprint to a uCDN, the original intended coverage of the CDN might not represent its actual value to the federation of CDNs. Consider an ISP-A and ISP-B that both field their own CDNs, which they federate through CDNi. A given user E, who is customer of ISP-B, might happen to be topologically closest to a cache fielded by ISP-A, if E happens to live in a region where ISP-B has few customers and ISP-A has many. In this case, should ISP-A's CDN "cover" E? If ISP-B's CDN has a failure condition, should the uCDN understand that ISP-A's caches are potentially available back-ups - and if so, how does ISP-A advertise itself as a "standby" for E? What about the case where CDNs advertising to the same uCDN express overlapping coverage (for example, a federation mixing global and limited CDNs)?

The answers to these questions greatly depend on how much information

we want the uCDN to use to make a selection of a dCDN. If a uCDN has three dCDNs to choose from that "cover" the IP address of user E, obviously the uCDN might be interested to know how optimal the coverage is from each of the dCDNs - coverage need not be binary, either provided or not provided. dCDNs could advertise a coverage "score," for example, and provided that they all reported scores fairly on the same scale, uCDNs could use that to make their topological optimality decision. Alternatively, dCDNs could for their footprint advertise the IP addresses of their caches rather than prefix "coverage," and let the uCDN decide for itself (based on its own topological intelligence) which dCDN has better resources to serve a given user.

3.2. Capabilities and Dynamic Data

In cases where the apparent footprint of dCDNs overlaps, uCDNs might also want to rely on a host of other factors to evaluate the respective merits of dCDNs. These include facts related to the caches themselves, to the network where the cache is deployed, to the nature of the resource sought and to the administrative policies of the respective networks.

In the absence of network-layer impediments to reaching caches, the choice to limit coverage is necessarily an administrative policy. Much policy must be agreed upon before CDNs can merge into federations, including questions of membership, compensation, volumes and so on. A uCDN certainly will factor these sorts of considerations into its decision to select a dCDN, but there is probably little need for dCDNs to actually advertise them through an interface - they will be settled out of band as a precondition for federating.

Other facts about the dCDN would be expressed through the interface to the uCDN. Some capabilities of a dCDN are static, and some are highly dynamic. Expressing the total storage built into its caches, for example, changes relatively rarely, whereas the amount storage in use at any given moment is highly volatile. Network bandwidth similarly could be expressed as either total bandwidth available to a cache, or based on the current state of the network. A cache may at one moment lack a particular resource in storage, but have it the next.

The semantics of the capabilities interface will depend on how much of the dCDN state needs to be pushed to the uCDN in order for it to make a decision.

3.3. Advertisement versus Queries

In a federated CDN environment, each dCDN shares some of its state with the uCDN, which the uCDN uses to build a unified picture of all of the dCDNs available to it. In architectures that share detailed capability information, the uCDN could basically perform the entire request-routing intelligence down to selecting a particular cache before sending the request to the dCDN (note that within the current CDNI WG scope, such direct selection of specific caches by the uCDN is out of scope). However, when the uCDN must deal with many potential dCDNs, this approach does not scale. Especially as CDNs scale up from dozens or hundreds of caches to thousands or tens of thousands, the volume of updates to footprint and capability may become onerous.

Were the volume of updates to exceed the volumes of requests to the uCDN, it might make more sense for the uCDN to query dCDNs upon receiving requests, instead of receiving advertisements and tracking the state of dCDNs itself. The advantage of querying dCDNs would be that much of the dynamic data that dCDNs cannot share with the uCDN would now be factored into the uCDN's decision. dCDNs need not replicate any state to the uCDN - uCDNs could effectively operate in a stateless mode.

The semantics of both footprint and capability advertisement depend on the service model here: are there cases where a synchronous query/response model would work better for the uCDN decision than a state replication model?

3.4. Avoiding or Handling 'cheating' Downstream CDNs

In a situation where more than one dCDN is willing to serve a given end user request, it might be attractive for a dCDN to "cheat" in the sense that the dCDN provides inaccurate information to the uCDN in order to convince the uCDN to select it opposed to "competing" other dCDNs. It is therefore desirable to take away the incentive for dCDNs to cheat (in information advertised) as much as possible. One option here is to make the information the dCDN advertises somehow verifiable for the uCDN. On the other hand, a cheating dCDN might be avoided or handled by the fact that there will be strong contractual agreements between a uCDN and a dCDN, so that a dCDN would risk severe penalties or legal consequences when caught cheating.

Overall, it seems that information a dCDN advertises should (in the long run) be somehow verifiable by the uCDN. However, it is probably an overly strict requirement to demand that such verification must be possible "immediately", i.e. during the request routing process

itself. If the uCDN can detect a cheating dCDN at a later stage, it should suffice for the dCDN to "de-incentivize" cheating because it would negatively affect long-term business relationships with uCDNs.

3.5. Focus on Main Use Cases may Simplify Things

To narrow down semantics for "footprint" and "capabilities" in the CDNI context, it can be useful to initially focus on key use cases to be addressed by the CDNI WG that are to be envisioned the main deployments in the foreseeable future. In this regard, a main realistic use case is the existence of ISP-owned CDNs, which essentially cover a certain operator's network. At the same time, however, the possibility of overlapping footprints should not be excluded, i.e. the scenario where more than one dCDN claims it can serve a given end user request. Further, it seems reasonable to assume that in most use cases it is the uCDN that makes the decision on selecting a certain dCDN for request routing based on information the uCDN has received from this particular dCDN.

4. Towards Semantics for Footprint Advertisement

Based on the characterizations in existing documents (see Section 2), we can distill some "rough" candidates for definitions of a footprint:

- o Footprint could be defined by "layer-3 coverage", where coverage refers to a set of prefixes, a geographic region, or similar boundary.
- o Footprint could alternatively be defined as "a class of end user requests a dCDN is willing to serve".

Independent of the exact definition of footprint, a footprint likely needs to be able to include the connectivity of a given dCDN to other CDNs that may be able to serve content to users on behalf of that dCDN, to cover cases where there is a transitive CDN interconnection. Further, the downstream CDN must be able to express its footprint to an interested upstream CDN (uCDN) in a comprehensive form, e.g., as a complete data set containing the complete footprint. Making incremental updates, however, to express dynamic changes in state is also desirable.

Different concrete candidates for a footprint are imaginable (set of prefixes, a geographic region, ...). Among these, "set of IP-prefixes" may potentially be a useful footprint in the CDDNI context. Such footprint information must be able to contain full IP addresses (i.e., a /32 for IPv4 and a /128 for IPv6) and also IP prefixes with an arbitrary prefix length. There must be support for multiple IP address version, i.e., IPv4 and IPv6 in the footprint.

Roughly speaking, footprint can be defined as "willingness to serve" by a downstream CDN. However, in addition to simply "willingness to serve", the uCDN needs to have some additional information to make a dCDN selection decision. The uCDN needs additional information so that it can assess "how well" a given dCDN can actually serve a given end user request. One can imagine that such additional information is implicitly associated with a given footprint, e.g. due to (from the request routing interface's perspective out-of-band) contractual agreements (e.g. SLAs), business relationships, or perceived dCDN quality in the past. As an alternative, such additional information could also be explicitly be tagged along with the footprint.

5. Towards Semantics for Capabilities Advertisement

The dCDN must be able to express its general capabilities to the uCDN. These general capabilities could express if the dCDN supports a given service, for instance, WWW delivery, Video on Demand (VoD) delivery based on flash or apple technologies, or live streaming based on RTSP.

The dCDN must be able to express particular capabilities for the delivery in a particular footprint area. For example, the dCDN might in general offer VoD but not in some areas, either for maintenance reasons or because this particular area cannot deliver this type of service. In such cases, i.e. where capabilities must be expressed on a per footprint basis, it may be beneficial to combine footprint and capabilities advertisement.

High-level and very rough semantics for (and characteristics of) capabilities are thus:

- o Capabilities are types of information that allow a uCDN to determine if a downstream CDN is able (and willing) to accept the delegated content request.
- o Some capabilities may change dynamically based on the state of the network or a cache.

Capabilities seem to fall into several broad categories. Some are capabilities associated with a resource itself, like the codecs or streaming technologies in which a particular resource is available. Some capabilities are associated with the cache: these include the load state, available storage resources, and so on. Some capabilities are associated with the network where the cache is deployed, including available bandwidth for streaming, availability of QoS mechanisms, and so on.

Some capabilities reflect administrative restrictions or policies that may affect the decisions made up a uCDN. It seems unlikely these factors will be shared through the interface, however.

Cache capabilities are:

- o aggregate information (i.e. not for individual caches, but still helpful for dCDN selection) about load, or "excessive load"
- o aggregate information (i.e. not for individual caches, but still helpful for dCDN selection) about available resources, storage resources

- o aggregate information (i.e. not for individual caches, but still helpful for dCDN selection) regarding failure conditions

Resource Capabilities:

- o supported range of playback devices
- o supported range of delivery technologies
- o specific delivery protocols
- o supported content types (MIME)

Network Capabilities:

- o meeting a certain QoS
- o distribution and delivery priority
- o streaming bandwidth

Outside the scope of capability advertisements are administrative capabilities, such as:

- o policy (membership in the federation, etc)
- o administrative limits, e.g. maximum aggregate volume of content delivered monthly
- o indication of distribution/delivery fees

6. Open Issues and Questions

The following open issues deserve discussion in the CDNI WG:

- o What is the service model of this interface: Does the uCDN always query the dCDNs? Or does the dCDN always push information to the uCDNs?
- o Should footprint advertisement be based on prefixes, on cache addresses, or on other facts?
- o Should capability advertisement include only static attributes of the CDN, or should it factor in dynamic attributes as well?
- o How can the dCDN express the associated costs of delivery, either monetary or virtual costs, to the uCDN for the actual delivery? Is this in scope of this interface?
- o Are monetary delivery costs explicitly in scope of capabilities advertisement?
- o Does a footprint need to include the "reachability" of the dCDN to other CDNs that may be able to serve content to users on behalf of dCDN?
- o What is the assumed business relationship between the uCDN and the dCDN? Is the uCDN always the "authoritative" CDN provider which transitively has itself contracted several downstream CDN providers?

7. Security Considerations

Security considerations will be discussed in a future version of this document.

8. Conclusion

This document tries to capture the semantics of the "Footprint and Capabilities Advertisement" part of the CDNI Request Routing interface, i.e. the desired meaning and what "Footprint and Capabilities Advertisement" is expected to offer within CDNI, also reflecting ongoing discussions in the CDNI design team on "Footprint and Capabilities Advertisement". Several key design decisions and open questions have been discussed.

The discussion in this document has the objective to facilitate the choosing of one or more suitable protocols for "Footprint and Capabilities Advertisement" within CDNI Request Routing. It is the goal of this document to capture the outcome of further progress of the CDNI WG and the design team on "Footprint and Capabilities Advertisement" including answers to currently open questions in future versions of this draft.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

- [I-D.davie-cdni-framework]
Davie, B. and L. Peterson, "Framework for CDN Interconnection", draft-davie-cdni-framework-01 (work in progress), October 2011.
- [I-D.ietf-cdni-problem-statement]
Niven-Jenkins, B., Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", draft-ietf-cdni-problem-statement-08 (work in progress), June 2012.
- [I-D.ietf-cdni-requirements]
Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", draft-ietf-cdni-requirements-03 (work in progress), June 2012.
- [I-D.ietf-cdni-use-cases]
Bertrand, G., Emile, S., Burbridge, T., Eardley, P., Ma, K., and G. Watson, "Use Cases for Content Delivery Network Interconnection", draft-ietf-cdni-use-cases-09 (work in progress), July 2012.
- [I-D.peterson-cdni-strawman]
Peterson, L. and J. Hartman, "A Simple Approach to CDN Interconnection", draft-peterson-cdni-strawman-01 (work in progress), May 2011.
- [I-D.previdi-cdni-footprint-advertisement]
Previdi, S., Faucheur, F., Faucheur, F., Medved, J., and L. Faucheur, "CDNI Footprint Advertisement", draft-previdi-cdni-footprint-advertisement-01 (work in progress), March 2012.
- [I-D.xiaoyan-cdni-requestrouting]
He, X., Li, J., Dawkins, S., and G. Chen, "Request Routing for CDN Interconnection", draft-xiaoyan-cdni-requestrouting-01 (work in progress), June 2011.

Appendix A. Acknowledgment

Jan Seedorf is partially supported by the COAST project (Content Aware Searching, retrieval and sTreaming, <http://www.coast-fp7.eu>), a research project supported by the European Commission under its 7th Framework Program (contract no. 248036). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the COAST project or the European Commission.

Martin Stiernerling provided initial input to this document and valuable comments to the ongoing discussions among the authors of this document.

Authors' Addresses

Jan Seedorf
NEC
Kurfuerstenanlage 36
Heidelberg 69115
Germany

Phone: +49 6221 4342 221
Fax: +49 6221 4342 155
Email: seedorf@neclab.eu

Jon Peterson
NeuStar
1800 Sutter St Suite 570
Concord CA 94520
USA

Phone:
Fax:
Email: jon.peterson@neustar.biz

Stefano Previdi
Cisco Systems
Via Del Serafico 200
Rome 0144
Italy

Phone:
Fax:
Email: sprevidi@cisco.com

CDNI
Internet-Draft
Intended status: Informational
Expires: August 29, 2013

J. Seedorf
NEC
J. Peterson
Neustar
S. Previdi
Cisco
R. van Brandenburg
TNO
K. Ma
Azuki Systems, Inc.
February 25, 2013

CDNI Request Routing: Footprint and Capabilities Semantics
draft-spp-cdni-rr-foot-cap-semantics-04

Abstract

This document tries to capture the semantics of the "Footprint and Capabilities Advertisement" part of the CDNI Request Routing interface, i.e. the desired meaning and what "Footprint and Capabilities Advertisement" is expected to offer within CDNI. The discussion in this document has the goal to facilitate the choosing of one or more suitable protocols for "Footprint and Capabilities Advertisement" within CDNI Request Routing.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 29, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction and scope	3
2. CDNI FCI in existing CDNI Documents	4
3. Design Decisions for Footprint and Capabilities	7
3.1. Advertising Limited Coverage	7
3.2. Capabilities and Dynamic Data	8
3.3. Advertisement versus Queries	9
3.4. Avoiding or Handling 'cheating' dCDNs	9
3.5. Focus on Main Use Cases may Simplify Things	10
4. Main Use Case to foster the Clarification of Semantics	11
5. Towards Semantics for Footprint Advertisement	12
6. Towards Semantics for Capabilities Advertisement	14
7. Open Issues and Questions	17
8. Security Considerations	18
9. References	19
9.1. Normative References	19
9.2. Informative References	19
Appendix A. Acknowledgment	20
Authors' Addresses	21

1. Introduction and scope

The CDNI working group is working on a set of protocols to enable the interconnection of multiple CDNs to a CDN federation. This CDN-federation should serve multiple purposes, as discussed in [RFC6770], for instance, to extend the reach of a given CDN to areas in the network which are not covered by this particular CDN.

The goal of this document is to achieve a clear understanding in the CDNI WG about the semantics associated with the CDNI Request Routing Footprint & Capabilities Advertisement Interface (from now on referred to as FCI), in particular the type of information a downstream CDN 'advertises' regarding its footprint and capabilities. To narrow down undecided aspects of these semantics, this document tries to establish a common understanding of what the FCI should offer and accomplish in the context of CDN Interconnection.

It is explicitly outside the scope of this document to decide on specific protocols to use for the FCI.

General assumptions in this document:

- o The CDNs participating in the CDN federation have already performed a boot strap process, i.e., they have connected to each other, either directly or indirectly, and can exchange information amongst each other.
- o The uCDN has received footprint and/or capability advertisements from a set of dCDNs. Footprint advertisement and capability advertisement need not use the same underlying protocol.
- o The upstream CDN (uCDN) receives the initial request-routing request from the endpoint requesting the resource.

This document is organized as follows. First, a recap of the definition of "footprint and capabilities advertisement" in existing documents is given, attempting to distill the apparent common understanding of what the terms 'footprint' and 'capabilities' mean in the context of CDNI. Then, the detailed semantics of the footprint advertisement mechanism and the capability advertisement mechanism will be discussed. Finally, open issues and questions to be discussed in the CDNI WG will be listed.

2. CDNI FCI in existing CDNI Documents

In the following section, the existing descriptions of the CDNI FCI interface in existing documents will be examined. After this, the apparent common understanding of what this interface is intended to offer and accomplish will be carved out.

The CDNI Problem Statement [RFC6707] describes footprint and capabilities advertisement as: "[enabling] a Request Routing function in an Upstream CDN to query a Request Routing function in a Downstream CDN to determine if the Downstream CDN is able (and willing) to accept the delegated Content Request". In addition, the draft says "the CDNI Request Routing interface is also expected to enable a downstream CDN to provide to the upstream CDN (static or dynamic) information (e.g. resources, footprint, load) to facilitate selection of the downstream CDN by the upstream CDN request routing system when processing subsequent content requests from User Agents". It thus considers "resources" and "load" as capabilities to be advertised by the downstream CDN.

The CDNI Use Cases document [RFC6770] describes capabilities as "... supported range of devices and User Agents or the supported range of delivery technologies". Examples for such capabilities given are specific delivery protocols, technology migration, and meeting a certain QoS.

The CDNI requirements draft [I-D.ietf-cdni-requirements] lists several requirements relevant for the "footprint and capabilities advertisement" part of the CDNI request routing interface. In summary, the following requirements for the CDNI Request Routing Interface and general requirements are relevant for the understanding of the semantics of "footprint and capabilities advertisement":

- o GEN-4 [HIGH], "The CDNI solution shall not require intra-CDN information to be exposed to other CDNs for effective and efficient delivery of the content. Examples of intra-CDN information include surrogate topology, surrogate status, cached content, etc."
- o GEN-9 [MED], "The CDNI solution should support cascaded CDN redirection (CDN1 redirects to CDN2 that redirects to CDN3) to an arbitrary number of levels beyond the first level."
- o GEN-10 [MED], "The CDNI solution should support an arbitrary topology of interconnected CDNs (i.e. the CDN topology cannot be restricted to a tree, a loop-free topology, etc.)."

- o GEN-11 [HIGH], "The CDNI solution shall prevent looping of any CDNI information exchange."
- o REQ-1 [HIGH], allowing the downstream CDN "to communicate to the Upstream CDN coarse information about the Downstream CDN ability and/or willingness to handle requests from the Upstream CDN. For example, this could potentially include a binary signal ("Downstream CDN ready/not-ready to take additional requests from Upstream CDN") to be used in case of excessive load or failure condition in the Downstream CDN."
- o REQ-2 [MED], allowing the downstream CDN to communicate capabilities such as supported content types and delivery protocols, a set of metrics/attributes (e.g. Streaming bandwidth, storage resources, distribution and delivery priority), a set of affinities (e.g. Preferences, indication of distribution/delivery fees), information to facilitate request redirection, as well as footprint information (e.g. "layer-3 coverage").
- o REQ-3 [MED], "In the case of cascaded redirection, the CDNI Request-Routing interface shall allow the Downstream CDN to also include in the information communicated to the Upstream CDN, information on the capabilities, resources and affinities of CDNs to which the Downstream CDN may (in turn) redirect requests received by the Upstream CDN. In that case, the CDNI Request-Routing interface shall prevent looping of such information exchange."
- o REQ-4 [LOW], allowing the downstream CDN to communicate "aggregate information on CDNI administrative limits and policy" (e.g. the maximum number of requests redirected by the Upstream CDN to be served simultaneously by the Downstream CDN or maximum aggregate volume of content (e.g. in Terabytes) to be delivered by the Downstream CDN over a time period).
- o REQ-11 [LOW], "The CDNI Request-Routing protocol may support a mechanism allowing an Upstream CDN to avoid redirecting a request to a Downstream CDN if that is likely to result in the total redirection time exceeding some limit."

Note that in REQ-2 [MED] "Layer-3 coverage" is given as an example of what "footprint" information might convey in the CDNI requirements draft [I-D.ietf-cdni-requirements]. Also, note that REQ-3 [MED] addresses cascaded (transitive) downstream CDNs. In such a case, a downstream CDN needs to include (in its advertisement information that it conveys to an upstream CDN) aggregate footprint and capabilities information for any further transitive downstream CDNs. Such information may be included implicitly (i.e. the cascaded dCDN

is oblivious to the uCDN), or explicitly (i.e. the cascaded dCDN of the fact that there is a cascaded dCDN is visible to the uCDN). In either case, logic is needed to process incoming footprint information from a cascaded dCDN and decide if/how it is to be re-advertised/aggregated when advertising footprint to an upstream CDN.

The CDNI framework draft [I-D.ietf-cdni-framework] describes a "footprint" as in [I-D.previdi-cdni-footprint-advertisement], consisting of two parts: 1) "a class of end user requests (represented, for example, by a set of IP prefixes, or a geographic region) that the dCDN is willing and able to serve directly, without use of another dCDN", and 2) "the connectivity of the dCDN to other CDNs that may be able to serve content to users on behalf of dCDN". The term "connectivity" has recently been replaced with "reachability" in [I-D.previdi-cdni-footprint-advertisement], and as discussed above, "without use of another dCDN" may include aggregated transitive dCDNs. Further examples for capabilities are "the ability to handle certain types of content (e.g. specific streaming formats) or quality of service (QoS)." Content handling capabilities discussed in [I-D.ma-cdni-capabilities] include delivery and acquisition protocols, redirection modes, and metadata related capabilities (e.g., authorization algorithm).

From reading the various draft listed above, it is safe to conclude that neither the term 'footprint' nor 'capabilities' has been clearly and unambiguously defined in these documents and a very broad range of potential capabilities is listed.

3. Design Decisions for Footprint and Capabilities

A large part of the difficulty in discussing the FCI lies in understanding what exactly is meant when trying to define footprint in terms of "coverage" or "reachability." While the operators of CDNs pick strategic locations to situate caches, a cache with a public IPv4 address is reachable by any endpoint on the Internet unless some policy enforcement precludes the use of the cache.

Some CDNs aspire to cover the entire world, which we will henceforth call global CDNs. The footprint advertised by such a CDN in the CDNI environment would, from a coverage or reachability perspective, presumably cover all prefixes. Potentially more interesting for CDNI use cases, however, are CDNs that claim a more limited coverage, but seek to federate with other CDNs in order to create a single CDN fabric which shares resources.

Futhermore, not all capabilities need be footprint restricted. Depending upon the use case, the optimal semantics of "footprints with capability attributes" vs. "capabilities with footprint restrictions" are not clear.

The key to understanding the semantics of footprint and capability advertisement lies in understand why a dCDN would advertise a limited coverage area, and how a uCDN would use such advertisements to decide among one of several dCDNs. The following section will discuss some of the trade-offs and design decisions that need to be decided upon for the CDNI FCI.

3.1. Advertising Limited Coverage

The basic use case that would motivate a dCDN to advertise a limited coverage is that the CDN was built to cover only a particular portion of the Internet. For example, an ISP could purpose-build a CDN to serve only their own customers by situating caches in close topological proximity to high concentrations of their subscribers. The ISP knows the prefixes it has allocated to end users and thus can easily construct a list of prefixes that its caches were positioned to serve.

When such a purpose-built CDN joins a federation, however, and advertises its footprint to a uCDN, the original intended coverage of the CDN might not represent its actual value to the federation of CDNs. Consider an ISP-A and ISP-B that both field their own CDNs, which they federate through CDNI. A given user E, who is customer of ISP-B, might happen to be topologically closest to a cache fielded by ISP-A, if E happens to live in a region where ISP-B has few customers and ISP-A has many. In this case, should ISP-A's CDN "cover" E? If

ISP-B's CDN has a failure condition, should the uCDN understand that ISP-A's caches are potentially available back-ups - and if so, how does ISP-A advertise itself as a "standby" for E? What about the case where CDNs advertising to the same uCDN express overlapping coverage (for example, a federation mixing global and limited CDNs)?

The answers to these questions greatly depend on how much information we want the uCDN to use to make a selection of a dCDN. If a uCDN has three dCDNs to choose from that "cover" the IP address of user E, obviously the uCDN might be interested to know how optimal the coverage is from each of the dCDNs - coverage need not be binary, either provided or not provided. dCDNs could advertise a coverage "score," for example, and provided that they all reported scores fairly on the same scale, uCDNs could use that to make their topological optimality decision. Alternatively, dCDNs could for their footprint advertise the IP addresses of their caches rather than prefix "coverage," and let the uCDN decide for itself (based on its own topological intelligence) which dCDN has better resources to serve a given user.

In summary, the semantics of advertising footprint depend on whether such qualitative metrics for expressing footprint (such as the coverage 'score' mentioned above) should be part of the CDNI FCI, or if it should focus just on 'binary' footprint.

3.2. Capabilities and Dynamic Data

In cases where the apparent footprint of dCDNs overlaps, uCDNs might also want to rely on a host of other factors to evaluate the respective merits of dCDNs. These include facts related to the caches themselves, to the network where the cache is deployed, to the nature of the resource sought and to the administrative policies of the respective networks.

In the absence of network-layer impediments to reaching caches, the choice to limit coverage is necessarily an administrative policy. Much policy must be agreed upon before CDNs can merge into federations, including questions of membership, compensation, volumes and so on. A uCDN certainly will factor these sorts of considerations into its decision to select a dCDN, but there is probably little need for dCDNs to actually advertise them through an interface - they will be settled out of band as a precondition for federating.

Other facts about the dCDN would be expressed through the interface to the uCDN. Some capabilities of a dCDN are static, and some are highly dynamic. Expressing the total storage built into its caches, for example, changes relatively rarely, whereas the amount storage in

use at any given moment is highly volatile. Network bandwidth similarly could be expressed as either total bandwidth available to a cache, or based on the current state of the network. A cache may at one moment lack a particular resource in storage, but have it the next.

The semantics of the capabilities interface will depend on how much of the dCDN state needs to be pushed to the uCDN and qualitatively how often that information should be updated.

3.3. Advertisement versus Queries

In a federated CDN environment, each dCDN shares some of its state with the uCDN, which the uCDN uses to build a unified picture of all of the dCDNs available to it. In architectures that share detailed capability information, the uCDN could basically perform the entire request-routing intelligence down to selecting a particular cache before sending the request to the dCDN (note that within the current CDNI WG scope, such direct selection of specific caches by the uCDN is out of scope). However, when the uCDN must deal with many potential dCDNs, this approach does not scale. Especially as CDNs scale up from dozens or hundreds of caches to thousands or tens of thousands, the volume of updates to footprint and capability may become onerous.

Were the volume of updates to exceed the volumes of requests to the uCDN, it might make more sense for the uCDN to query dCDNs upon receiving requests (as is the case in the recursive redirection mode described in [I-D.ietf-cdni-framework]), instead of receiving advertisements and tracking the state of dCDNs itself. The advantage of querying dCDNs would be that much of the dynamic data that dCDNs cannot share with the uCDN would now be factored into the uCDN's decision. dCDNs need not replicate any state to the uCDN - uCDNs could effectively operate in a stateless mode.

The semantics of both footprint and capability advertisement depend on the service model here: are there cases where a synchronous query/response model would work better for the uCDN decision than a state replication model?

3.4. Avoiding or Handling 'cheating' dCDNs

In a situation where more than one dCDN is willing to serve a given end user request, it might be attractive for a dCDN to 'cheat' in the sense that the dCDN provides inaccurate information to the uCDN in order to convince the uCDN to select it opposed to 'competing' dCDNs. It could therefore be desirable to take away the incentive for dCDNs to cheat (in information advertised) as much as possible. One option

here is to make the information the dCDN advertises somehow verifiable for the uCDN. On the other hand, a cheating dCDN might be avoided or handled by the fact that there will be strong contractual agreements between a uCDN and a dCDN, so that a dCDN would risk severe penalties or legal consequences when caught cheating.

Overall, it seems that information a dCDN advertises should (in the long run) be somehow qualitatively verifiable by the uCDN, though possibly through non-real-time out-of-band audits. It is probably an overly strict requirement to mandate that such verification be possible "immediately", i.e. during the request routing process itself. If the uCDN can detect a cheating dCDN at a later stage, it should suffice for the uCDN to "de-incentivize" cheating because it would negatively affect the long-term business relationship with a particular dCDN.

3.5. Focus on Main Use Cases may Simplify Things

To narrow down semantics for "footprint" and "capabilities" in the CDNI context, it can be useful to initially focus on key use cases to be addressed by the CDNI WG that are to be envisioned the main deployments in the foreseeable future. In this regard, a main realistic use case is the existence of ISP-owned CDNs, which essentially cover a certain operator's network. At the same time, however, the possibility of overlapping footprints should not be excluded, i.e. the scenario where more than one dCDN claims it can serve a given end user request. The ISPs may also choose to federate with a fallback global CDN.

It seems reasonable to assume that in most use cases it is the uCDN that makes the decision on selecting a certain dCDN for request routing based on information the uCDN has received from this particular dCDN. It may be assumed that 'cheating' CDNs will be dealt with via means outside the scope of CDNI and that the information advertised between CDNs is accurate. In addition, excluding the use of qualitative information (e.g., cache proximity, delivery latency, cache load) to predict the quality of delivery would further simplify the use case allowing it to better focus on the basic functionality of the FCI.

4. Main Use Case to foster the Clarification of Semantics

Focusing on a main use case that contains a simple (yet somewhat challenging), realistic, and generally imaginable scenario can help in narrowing down the requirements for the CDNI FCI. To this end, the following (simplified) use case can help in clarifying the semantics of footprint and capabilities for CDNI. In particular, the intention of the use case is to clarify what information needs to be exchanged on the CDNI FCI, what types of information need to be supported in a mandatory fashion (and which should be considered optional), and what types of information need to be updated with respect to a priori established CDNI contracts.

In short, one can imagine the following use case: A given uCDN has several dCDNs. It selects one dCDN for delivery protocol A and footprint 1 and another dCDN for delivery protocol B and footprint 1. The dCDN that serves delivery protocol B has a further, transitive (level-2) dCDN, that serves delivery protocol B in a subset of footprint 1 where the first-level dCDN cannot serve delivery protocol B itself. What happens if capabilities change in the transitive level-2 dCDN that might affect how the uCDN selects a level-1 dCDN (e.g. in case the level-2 dCDN cannot serve delivery protocol B anymore)? How will these changes be conveyed to the uCDN? In particular, what information does the uCDN need to be able to select a new first-level dCDN, either for all of footprint 1 or only for the subset of footprint 1 that the transitive level-2 dCDN served on behalf of the first-level dCDN?

5. Towards Semantics for Footprint Advertisement

Roughly speaking, "footprint" can be defined as "ability and willingness to serve" by a downstream CDN. However, in addition to simple "ability and willingness to serve", the uCDN may wish to have additional information to make a dCDN selection decision, e.g., "how well" a given dCDN can actually serve a given end user request. The "ability and willingness" to serve should be distinguished from the subjective qualitative measurement of "how well" it was served. One can imagine that such additional information is implicitly associated with a given footprint, e.g. due to contractual agreements (e.g. SLAs), business relationships, or perceived dCDN quality in the past. As an alternative, such additional information could also be explicitly tagged along with the footprint.

It is reasonable to assume that a significant part of the actual footprint advertisement will happen in contractual agreements between participating CDNs, i.e. prior to the advertisement phase using the CDNI FCI. The reason for this assumption is that any contractual agreement is likely to contain specifics about the dCDN coverage (i.e. the dCDN footprint) the contractual agreement applies to. In particular, additional information to judge the delivery quality associated with a given dCDN footprint might be defined in contractual agreements (i.e. outside of the CDNI FCI). Further, one can assume that dCDN contractual agreements about the delivery quality associated with a given footprint will probably be based on high-level aggregated statistics (i.e. not too detailed).

Given that a large part of footprint advertisement will actually happen in contractual agreements, the semantics of CDNI footprint advertisement refer to answering the following question: what exactly still needs to be advertised by the CDNI FCI? For instance, updates about temporal failures of part of a footprint can be useful information to convey via the CDNI request routing interface. Such information would provide updates on information previously agreed in contracts between the participating CDNs. In other words, the CDNI FCI is a means for a dCDN to provide changes/updates regarding a footprint it has prior agreed to serve in a contract with a uCDN.

Generally speaking, one can imagine two categories of footprint to be advertised by a dCDN:

- o Footprint could be defined based on (layer-3) "coverage/reachability", where coverage/reachability refers to a set of prefixes, a geographic region, or similar boundary. The dCDN claims that it can cover/reach 'end user requests coming from this footprint'.

- o Footprint could be defined based on "resources", where resources refers to surrogates/caches a dCDN claims to have (e.g., the location of surrogates/resources). The dCDN claims that 'from this footprint' it can serve incoming end user requests.

For each of these footprint types, there are capabilities associated with a given footprint, i.e. the capabilities (e.g., delivery protocol, redirection mode, metadata) supported in the coverage area for a "coverage/reachability" defined footprint, or the capabilities of resources (e.g., delivery protocol, redirection mode, metadata support) for a "resources" defined footprint. It seems clear that "coverage/reachability" types of footprint must be supported within CDNI. It needs to be decided, whether - in addition to "coverage/reachability" types - also "resource" types of footprint should be supported within CDNI.

Different concrete candidates for a "coverage/reachability" footprint are imaginable. In particular, the following concrete types of footprint seem useful in the CDNI context: 'set of IP-prefixes', 'AS number list', and 'geographic region'. Among these, 'set of IP-prefixes' must be able to contain full IP addresses (i.e., a /32 for IPv4 and a /128 for IPv6) and also IP prefixes with an arbitrary prefix length. There must also be support for multiple IP address versions, i.e., IPv4 and IPv6, in the footprint.

Independent of the exact type of a footprint, a footprint might also include the connectivity of a given dCDN to other CDNs that may be able to serve content to users on behalf of that dCDN, to cover cases where there is a transitive CDN interconnection. Further, the downstream CDN must be able to express its footprint to an interested upstream CDN (uCDN) in a comprehensive form, e.g., as a complete data set containing the complete footprint. Making incremental updates, however, to express dynamic changes in state is also desirable.

6. Towards Semantics for Capabilities Advertisement

In general, the dCDN must be able to express its general capabilities to the uCDN. These general capabilities could express if the dCDN supports a given service, for instance, HTTP delivery, RTP/RTSP delivery or RTMP. Furthermore, the dCDN must be able to express particular capabilities for the delivery in a particular footprint area. For example, the dCDN might in general offer RTMP but not in some specific areas, either for maintenance reasons or because the caches covering this particular area cannot deliver this type of service. Hence, in certain cases footprint and capabilities are tied together and cannot be interpreted independently from each other. In such cases, i.e. where capabilities must be expressed on a per footprint basis, it may be beneficial to combine footprint and capabilities advertisement.

A high-level and very rough semantic for capabilities is thus the following: Capabilities are types of information that allow a uCDN to determine if a downstream CDN is able (and willing) to accept (and properly handle) a delegated content request. In addition, Capabilities are characterized by the fact that this information may possibly change over time based on the state of the network or caches.

At a first glance, several broad categories of capabilities seem useful to convey via an advertisement interface (and indeed many such candidate capabilities have been suggested in CDNI drafts, see Section 2). However, advertising capabilities that change highly dynamically (e.g. real-time delivery performance metrics, CDN resource load, or other highly dynamically changing QoS information) should probably not be in scope for the CDNI FCI. First, out of the multitude of possible metrics and capabilities, it is hard to agree on a subset and the precise metrics to be used. Second, and perhaps more importantly, it seems not feasible to specify such highly dynamically changing capabilities and the corresponding metrics within the CDNI charter time-frame.

Useful capabilities refer to information that does not change highly dynamically and which in many cases is absolutely necessary to decide on a particular dCDN for a given end user request. For instance, if an end user request concerns the delivery of a video file with a certain protocol (e.g. RTMP), the uCDN needs to know if a given dCDN has the capability of supporting this delivery protocol.

Similar to footprint advertisement, it is reasonable to assume that a significant part of the actual (resource) capabilities advertisement will happen in contractual agreements between participating CDNs, i.e. prior to the advertisement phase using the CDNI FCI. The role

of capability advertisement is hence rather to enable the dCDN to update a uCDN on changes since a contract has been set up (e.g. in case a new delivery protocol is suddenly being added to the list of supported delivery protocols of a given dCDN, or in case a certain delivery protocol is suddenly not being supported anymore due to failures). Capabilities advertisement thus refers to conveying information to a uCDN about changes/updates of certain capabilities with respect to a given contract.

Given these semantics, it needs to be decided what exact capabilities are useful and how these can be expressed. Since the details of CDNI contracts are not known at the time of this writing (and the CDNI interface should probably be agnostic to these contracts anyway), it remains to be seen what capabilities will be used to define agreements between CDNs in practice. One implication for standardization may be to initially only specify a very limited set of mandatory capabilities for advertisement and have on top of that a flexible data model that allows exchanging additional capabilities when needed. Still, agreement needs to be found on which capabilities (if any) should be mandatory among CDNs. As discussed in Section 3.5, finding the concrete answers to these questions can benefit from focusing on a small number of key use cases that are highly relevant and contain enough complexity to help in understanding what concrete capabilities are needed to facilitate CDN Interconnection.

Under the above considerations, the following capabilities seem useful as 'base' capabilities, i.e. ones that are needed in any case and therefore constitute mandatory capabilities to be supported by the CDNI FCI:

- o Delivery Protocol (e.g., HTTP vs. RTMP)
- o Acquisition Protocol (for acquiring content from a uCDN)
- o Redirection Mode (e.g., DNS Redirection vs. HTTP Redirection as discussed in [I-D.ietf-cdni-framework])
- o Capabilities related to CDNI Logging (e.g., supported logging mechanisms)
- o Capabilities related to CDNI Metadata (e.g., authorization algorithms or support for proprietary vendor metadata)

It is not feasible to enumerate all the possible options for the mandatory capabilities listed above (e.g., all the potential delivery protocols or metadata options) or anticipate all the future needs for additional capabilities. It would be unreasonable to burden the CDNI

FCI specification with defining each supported capability. The CDNI FCI specification should define a generic protocol for conveying any capability information. It would be reasonable to define a registry which initially contains the mandatory capabilities listed above, but may be extended as needs dictate. The CDNI FCI specification SHOULD define the registry (and the rules for adding new entries to the registry) for the different capability types. Each capability type MAY further have a list of valid values. The individual CDNI interface specifications which define a given capability SHOULD define any necessary registries (and the rules for adding new entries to the registry) for the values advertised for a given capability type.

The mandatory capabilities listed above generally relate to information that is configured on a content asset or group of assets basis via CDNI metadata. The capability requirements for acquisition and delivery protocol, redirection mode, and other mandatory metadata capabilities (e.g. authorization algorithms) are defined in [I-D.ietf-cdni-metadata].

Note: CDNI interface support for logging configuration (i.e., control interface vs. metadata interface) has not yet been decided. Once it has been decided, the corresponding CDNI interface specification should define the associated capability requirements.

7. Open Issues and Questions

The following open issues deserve further discussion in the CDNI WG:

- o What is the service model of this interface: Does the uCDN always query the dCDNs? Or does the dCDN always push information to the uCDNs?
- o In addition to "reachability" types of footprint, should also "resource" types of footprint be considered by CDNI (e.g., the location of surrogates/resources a dCDN has)?
- o Does a footprint need to explicitly include the "transitive reachability" of a dCDN to further dCDNs that may be able to serve content to users on behalf of dCDN?
- o What is the assumed business relationship between the uCDN and the dCDN? Is the uCDN always the "authoritative" CDN provider which transitively has itself contracted several downstream CDN providers?
- o How exactly can a given dCDN derive its footprint?
- o Should the footprint/capabilities advertisement interface only signal the delta with respect to a given contract (between a uCDN and a dCDN) or send the whole dCDN state each time?

8. Security Considerations

Security considerations will be discussed in a future version of this document.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

- [I-D.ietf-cdni-framework]
Peterson, L. and B. Davie, "Framework for CDN Interconnection", draft-ietf-cdni-framework-03 (work in progress), February 2013.
- [I-D.ietf-cdni-metadata]
Niven-Jenkins, B., Murray, R., Watson, G., Caulfield, M., Leung, K., and K. Ma, "CDN Interconnect Metadata", draft-ietf-cdni-metadata-00 (work in progress), October 2012.
- [I-D.ietf-cdni-requirements]
Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", draft-ietf-cdni-requirements-05 (work in progress), February 2013.
- [I-D.ma-cdni-capabilities]
Ma, K., "Content Distribution Network Interconnection (CDNI) Capabilities Interface", draft-ma-cdni-capabilities-01 (work in progress), February 2013.
- [I-D.previdi-cdni-footprint-advertisement]
Previdi, S., Faucheur, F., Faucheur, F., Medved, J., and L. Faucheur, "CDNI Footprint Advertisement", draft-previdi-cdni-footprint-advertisement-02 (work in progress), September 2012.
- [RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", RFC 6707, September 2012.
- [RFC6770] Bertrand, G., Stephan, E., Burbridge, T., Eardley, P., Ma, K., and G. Watson, "Use Cases for Content Delivery Network Interconnection", RFC 6770, November 2012.

Appendix A. Acknowledgment

Jan Seedorf is partially supported by the CHANGE project (CHANGE: Enabling Innovation in the Internet Architecture through Flexible Flow-Processing Extensions, <http://www.change-project.eu/>), a research project supported by the European Commission under its 7th Framework Program (contract no. 257422). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the CHANGE project or the European Commission.

Jan Seedorf has been partially supported by the COAST project (Content Aware Searching, retrieval and sTreaming, <http://www.coast-fp7.eu/>), a research project supported by the European Commission under its 7th Framework Program (contract no. 248036). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the COAST project or the European Commission.

Martin Stiernerling provided initial input to this document and valuable comments to the ongoing discussions among the authors of this document.

Authors' Addresses

Jan Seedorf
NEC
Kurfuerstenanlage 36
Heidelberg 69115
Germany

Phone: +49 6221 4342 221
Fax: +49 6221 4342 155
Email: seedorf@neclab.eu

Jon Peterson
NeuStar
1800 Sutter St Suite 570
Concord CA 94520
USA

Phone:
Fax:
Email: jon.peterson@neustar.biz

Stefano Previdi
Cisco Systems
Via Del Serafico 200
Rome 0144
Italy

Phone:
Fax:
Email: sprevidi@cisco.com

Ray van Brandenburg
TNO
Brassersplein 2
Delft 2612CT
The Netherlands

Phone: +31-88-866-7000
Email: ray.vanbrandenburg@tno.nl

Kevin J. Ma
Azuki Systems, Inc.
43 Nagog Park
Acton MA 01720
USA

Phone: +1 978-844-5100
Email: kevin.ma@azukisystems.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 10, 2013

E. Stephan
S. Ellouze
France Telecom - Orange
July 09, 2012

ALTO session for CDN Interconnection
draft-stephan-cdni-alto-session-ext-01

Abstract

The selection of a downstream CDN by an upstream CDN is based on multi-dimensional criteria such as the number of hops, the performance of the connections between the user-agent and downstream CDNs, the availability of downstream CDNs resources and business policies. Various protocols, such as BGP or ALTO, may be used by a downstream CDN to expose content routing information and interconnection preferences to an upstream CDN. This draft specifies the parameters of an ALTO session between two interconnected CDNs.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 10, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	4
1.1. Terminology	4
2. Use Cases	5
2.1. CDN1 views	8
2.2. CDN2 views	8
2.3. Map Maintenance	9
3. Requirements for an ALTO Session for CDNi	9
3.1. ALTO Information Customization	9
3.2. View HTTP GET	10
3.3. Initialization of the Session	10
3.4. Server Discovery	10
3.5. Maps Update	10
3.6. Information Resource Directory	11
3.7. Redistribution	11
3.8. PID Stability	11
3.9. Scalability	12
3.10. Performance	12
3.11. Heart Beat	13
3.12. dCDN Traffic Optimization	13
4. Specification of the ALTO Session for CDNi	13
4.1. ALTO session Framework	13
4.2. View Configuration	15
4.2.1. PoINT	15
4.2.2. View Configuration Examples	15
4.3. Session Configuration Parameters	16
4.4. Error Handling	17
5. Enhancements	17
5.1. Incremental Update	17
5.1.1. Level of Details of a Map	18
5.2. Bi-directional Exchange of Information	18
6. IANA Considerations	19
7. Security Considerations	19
8. Acknowledgments	19
9. References	20
9.1. Normative References	20
9.2. Informative References	20
Authors' Addresses	22

1. Introduction

The selection of a downstream CDN by an upstream CDN is based on multi-dimensional criteria such as the number of hops, the performance of the connections between the user-agent and downstream CDNs, the availability of downstream CDNs resources and business policies. Various protocols, such as BGP or ALTO, may be used by a downstream CDN to expose content routing information and interconnection preferences to an upstream CDN. This draft specifies the parameters of an ALTO session between two interconnected CDNs.

Currently the ALTO protocol is designed for the communication of network information to untrusted internet applications. In the context of a CDN interconnection (CDNi) there is a certain level of trust, at least enough to mount a subset of the interfaces depicted in [I-D.ietf-cdni-problem-statement]. In practice the level of trust differs with each interconnection. There are situations where a Cdni ALTO server has to exchange information with an ALTO client of an affiliate and with an ALTO client of a competitor (see [I-D.ietf-cdni-use-cases]).

In the first case topology hiding [RFC5693] may not be required. In the second case the operator of a dCDN may consider a fine control of the exposed information. Consequently the ALTO server of a dCDN operator must be able to adapt the information exposed to each uCDN.

The document discusses firstly the insightful aspects of such a use cases in section 2. Then in section 3 it presents the motivations for specifying an ALTO session to customize the information exposed in each CDN interconnection. In section 4, it provides a proposal for an appropriate specification of an ALTO session for a CDN interconnection. Finally it discusses different enhancements of interest to a Cdni ALTO session.

N.B.: this version of the memo covers only the Network Map.

1.1. Terminology

The reader must be familiar with the terminology given by the drafts [I-D.ietf-cdni-problem-statement], and [I-D.ietf-cdni-requirements], and [I-D.ietf-alto-protocol].

The following abbreviations are recalled:

dCDN : downstream CDN: The CDN which provide the delivery resource;

uCDN : upstream CDN: The CDN which may rely on dCDN server to deliver contents;

PID : Provider-defined Network Location Identifier;

NSP : Network Service Provider (e.g. ISP connecting End User to Internet);

ALTO Information Resource Directory: (Directory): The Information Resource Directory indicates to ALTO Clients which Information Resources are made available by an ALTO Server (section 7.6 [I-D.ietf-alto-protocol]).

Following are terms and abbreviations introduced in the document:

adCDN: ALTO downstream CDN server;

auCDN: ALTO upstream CDN client;

PIDs of Interest (PoINT) : The PIDs which are in the scope of an ALTO session or of a view. They may be defined as a list or by a XSLT-like statement (e.g. 'map/*/ipv6');

Costs of Interest (CoINT) : The Costs which are in the scope of an ALTO session or of a view;

ALTO Client-Server session: The logical association between an CDNi ALTO Client and an CDNi ALTO server which maintains the context across the ALTO HTTP connections made by the client to the server.

View: A view is the set of URIs which provide an auCDN with a mean for downloading the maps reflecting an agreement between an uCDN and a dCDN. A view is defined by PIDs of Interest (PoINT) and Costs of Interest (CoINT).

2. Use Cases

This section depicts a situation where a dCDN exposes information according to the agreements of each CDN interconnection. The information is exposed within an ALTO session based on the current ALTO protocol version. There is not time dependency between the content requests received by the upstream CDN and the information exchanged over the CDNi ALTO interface.

To ease the reading, the content of the Network Maps is intentionally limited with regards to real situation.

The use case is about a NSP which deployed a CDN named CDN0 over its network and where CDN0 acts as a downstream CDN for two CDNs named CDN1 and CDN2. CDN1 is an affiliate of CDN0.

In the figure 1, the network of NSP provides CDN0 with an aggregated view of the routing information. The grouping of the routing information results from the processing of information provided by BGP according to various policies of the NSP (network, content distribution, etc).

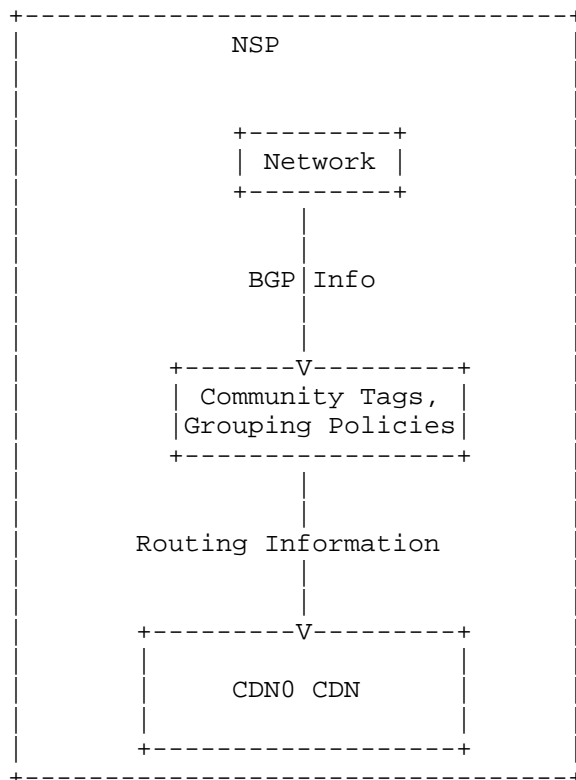


Figure 1: Internal Routing Information

The Figure 2 shows CDN0 acting as a dCDN for CDN1 and CDN2. CDN0 ALTO server (adCDN0) filters and sends stable Network and Cost Maps to the uCDN ALTO clients according to its policies and with respect to the peering agreement between the NSP and the operators of CDN1 and CDN2. adCDN0 is connected to 2 Cdni ALTO clients named auCDN1 and auCDN2.

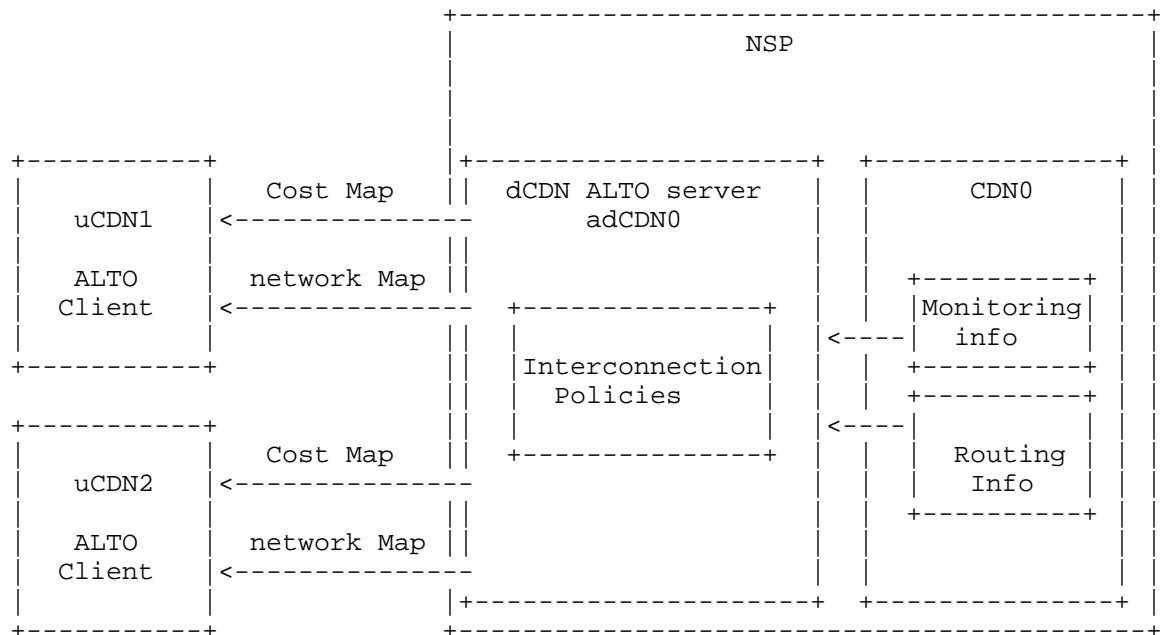


Figure 2: CDNs interconnection

The Figure 3 presents the internal representation of the Network Map computed by CDN0 ALTO server.

```

"map" : {
  "PID_DSL" : {
    "ipv4" : [
      "192.0.2.0/24",
      "198.51.100.0/25"
    ],
    "ipv6": [
      "2001:db8:0:1::/64",
    ]
  },
  "PID_FTTH" : {
    "ipv4" : [
      "198.51.100.128/25"
    ],
    "ipv6": [
      "2001:db8:0:2::/64"
    ]
  }
}

```

Figure 3: CDN0 internal Network Map

2.1. CDN1 views

CDN0 and CDN1 agreed that CDN1 needs only the IPv4 view of the Network Map. The Network Map, presented in figure 4, is downloadable by CDN1 at the URI

'<http://cdni.alto.example.com/CDN1/networkmap/ipv4>'.

```
"map" : {
  "PID_DSL" : {
    "ipv4" : [
      "192.0.2.0/24",
      "198.51.100.0/25"
    ]
  },
  "PID_FTTH" : {
    "ipv4" : [
      "198.51.100.128/25"
    ]
  }
}
```

Figure 4: CDN1 IPv4 Network Map view

2.2. CDN2 views

CDN0 and CDN2 have 2 separate agreements. Both are relative to the geographical extension of CDN2 coverage. The first agreement concerns the exposition of FTTH customers only. The second one covers IPv6 customers only. They are reflected as separated Network Maps. The first Network Map, exposed in figure 5, is downloadable by CDN2 at the URI '<http://cdni.alto.example.com/CDN2/networkmap/FTTH>'.

```
"map" : {
  "PID_FTTH" : {
    "ipv4" : [
      "198.51.100.128/25"
    ],
    "ipv6": [
      "2001:db8:0:2::/64"
    ]
  }
}
```

Figure 5: CDN2 FTTH Network Map.

The second Network Map, exposed in the figure 6, is downloadable by auCDN2 at the URI

'<http://cdni.alto.example.com/CDN2/networkmap/IPv6>'.

```
"map" : {  
  "PID_DSL" : {  
    "ipv6": [  
      "2001:db8:0:1::/64",  
    ]  
  },  
  "PID_FTTH" : {  
    "ipv6": [  
      "2001:db8:0:2::/64"  
    ]  
  }  
}
```

Figure 6: CDN2 IPv6 Network Map

2.3. Map Maintenance

auCDN1 and auCDN2 need a mean for maintaining the content of the maps. The ALTO server of CDN0 provides each view with an URI towards a list of the PIDs which were really modified in the last update. Each dCDN can download this information and determine whenever or not it have to update the Network Map of this view again.

This is not optimal. Nevertheless it provides an update mechanism based on HTTP GET which contribute to the reduction of both the volume of information exchanged and the processing on each side.

3. Requirements for an ALTO Session for Cdni

This section motivates the necessity of specifying an ALTO session between a dCDN and a uCDN with adequate features for addressing CDN interconnection requirements.

3.1. ALTO Information Customization

The current ALTO approach excludes that the Maps exposed by the ALTO server differ according to the client . This is enforced by section 7.2.6 of [I-D.ietf-alto-protocol] which recommends ignoring HTTP session parameters and HTTP cookies.

CDNi widely differs in such aspects because a dCDN operator must be able to adapt the information exposed to each uCDN according first to its policies and second to its agreements with uCDN. Moreover it is important for a uCDN client to optimize the volume and the relevance of the information received by avoiding downloading unwanted information in order to enhance the performance of the processing of the received data.

Consequently the customization of the ALTO interface between an uCDN and a dCDN requires the specification of a minimal set of session parameters. This must be specified inside the Cdni WG to provide a minimal level of interoperability amongst CDNs.

3.2. View HTTP GET

Currently [I-D.ietf-alto-protocol] (section 7.6.2 and 7.6.4.1) allows 2 Information Resources of the Information Resource Directory to match the same view of a map and to be downloadable using either an HTTP POST or a HTTP GET.

In the context of ALTO session for Cdni this is not allowed. A view of a map is accessible by a unique URI using HTTP GET request only. The session configuration defines all the information resources that an auCDN can download.

3.3. Initialization of the Session

The setting of the session between an uCDN and a dCDN reflects their agreements and expectancies. A minimal configuration of the session is required for ensuring an efficient initialization of the interface, for decreasing the service time, increasing the interoperability and improving the security.

The exchange of the session configuration parameters can be performed either out-of-band (human settings) or through the Cdni Control interface. In both cases the setting of a Cdni ALTO session requires an agreement between the 2 CDNs operators and a technical description of the session configuration (ALTO server and client addresses, URL, authentication methods, etc.), of the information which can be exchanged (PID of Interest, Cost of interest, level of details of the maps, etc) and of the way the information is exchanged (update method, time-scale for updates, etc).

3.4. Server Discovery

The discovery of a dCDN server by a uCDN relies on parameters exchanged out-of-band or on the Cdni Control interface. Consequently a CDN interconnection does not require any discovery mechanisms like described in [I-D.ietf-alto-server-discovery].

3.5. Maps Update

Maps update is under discussion in the ALTO WG as there is a strong need to optimize the volume of exchanged information and to reduce the duration of the acquisition of the updates. , [I-D.schwan-alto-incr-updates] presents solutions for incremental

update (download of update by the client). Another possibility is the specification of a synchronous update procedure where the server pushes the updates on the fly towards the client.

Incremental update and synchronous update may not be required for CDN interconnection when the customization of the session leads to a limited amount of information to be exchanged. Nevertheless, the adCDN server must at least provide hints to each auCDN for reducing the volume of exchanged information.

If no update mechanism is implemented, each view must include a information resource exposing a summary of the map updates (the list or the number of PIDs which were updated, etc). This approach provides the client logic with enough information to decide whether to re-download the whole map or not (e.g. depending on the importance of the PIDs which were updated).

3.6. Information Resource Directory

Section 7.6 of [I-D.ietf-alto-protocol] requires the availability of Information Resource Directory for exposing the Information Resources (i.e. URIs of the maps).

In a Cdni interconnection it is not necessary to provide such directory as the two interconnected CDNs previously agreed on the URI names. Besides, avoiding the exposition of URIs enhances the security of the system (see section 11.5. [I-D.ietf-alto-protocol]).

3.7. Redistribution

Redistribution of network Map and Cost Map by an uCDN is forbidden because first it results on the exposition of information exclusively destined to a well defined entity and second others uCDNs must not be flooded with unwished information.

The information exposed by dCDN to uCDN reflects only the agreement between the operators of these 2 CDNs only. Hence, redistributing maps content will lead to inconsistency because the semantic of the information differs with the session. As an example a PID name may cover different meaning or content.

3.8. PID Stability

Currently ALTO servers scramble the prefixes among the PIDs to prevent reverse engineering by ALTO clients ([I-D.ietf-alto-protocol], section 12.1).

CDNi situations differ widely in such aspects. Such nondeterministic semantics is totally unusable by a request routing function of a uCDN, or may lead to suboptimal decision. The dCDN must expose meaningful information to uCDN. Consequently the meaning of the PIDs must not change during the session duration.

As described in section 4.1 of [I-D.previdi-cdni-footprint-advertisement] a CDN acquires part of the content routing information from legacy BGP. As given by figure 1, The NSP may use part of the community tags carried by its legacy internal BGP to filter and gather the prefixes in stable groups (see section 5.1.7 of [I-D.ietf-alto-deployments]) that may then be used by its internal CDN [I-D.jenkins-alto-cdn-use-cases].

3.9. Scalability

The routing function of an uCDN might not require all the information that an ALTO server of an dCDN might expose. Furthermore, as by nature an uCDN interconnects with several dCDNs this volume might harm its performance and its reliability [I-D.ietf-alto-deployments].

The same applies for dCDN ALTO server. It must not be overloaded by dCDNs requests.

Consequently the Cdni ALTO session will provide dCDN and uCDN with a mean to shape the information to exchange in an interoperable manner. For instance, an uCDN may not want to receive the very last detailed level of the network map of all the dCDNs it is interconnected with; it may not want to receive each update; it may be interested only by one cost type attribute of the Cost Map service, etc.

N.B.: The situation will be even worse when the maps will include multi-cost as proposed by ([I-D.randriamasy-alto-multi-cost] and [I-D.marocco-alto-next] section 3.2) because the size of the maps will increase.

3.10. Performance

The amount of information to be processed impacts directly the performance of an auCDN. As an example an uCDN does not want to download all the PIDs when it needs only the PIDs of the Endpoints managed directly by each dCDN. Currently, as given by section 7.2.2. of [I-D.ietf-alto-protocol] , this is achieved using HTTP POST querying the ALTO Map Filtering Service or by HTTP GET of pre-generated maps.

To optimize the performance the ALTO Map Filtering Service is not exposed. Map filtering is accessible only through HTTP GET toward

pre-generated maps according to the configuration of the session agreed by the CDNs. Consequently the ALTO session configuration must include must include filters (PIDs, cost, etc) to reduce the volume of information exchanged about to the PIDs of Interest (PoINT) and to the Cost of Interest (CoINT) agreed by uCDN and dCDN operators. These filters apply during all the duration of the ALTO session.

3.11. Heart Beat

Neither the ALTO server nor the ALTO client want the session to be overflowed. A heart beat parameter is needed to control the intensity of the communication for exchanging information over the ALTO session. It provides a hint of the periodicity of the download of the maps by the client (e.g. every minute, hour, day, week, etc).

As an example to avoid useless maps download auCDN and adCDN might agreed to set the value of the heart beat to the period of refreshment of the considered maps.

3.12. dCDN Traffic Optimization

Considering that ALTO is about traffic optimization at the application level, in the context of a Cdni interconnection between an uCDN and a dCDN, ALTO is capable of covering the exchange of information from dCDN to uCDN, allowing for the optimization of the delivery at the uCDN side only. In contrast, exchanging information the other way around for allowing delivery optimization at the dCDN level is not addressed yet.

Indeed a dCDN is subject to rival uCDNs requesting resources based on information exposed by the dCDN. By exposing their constraints and their needs, the uCDNs requirements are better addressed by dCDNs through a smart resource provisioning and sharing.

A uCDN should be able to provide dCDN with information that may help dCDN to optimize it resources.

4. Specification of the ALTO Session for Cdni

This section specifies the ALTO session for Cdni.

4.1. ALTO session Framework

The figure 7 presents the Framework of the ALTO Session for Cdni interconnection:

The Map filtering logic is represented to reflect the customization of the content of the internal maps to the server according to the scope of the sessions with uCDNs.

There are PID and Cost filters to limit the scope of the session with regard to the content of the internal maps content of the server.

Network Map and Cost Map are unchanged. Nevertheless the session PIDs and Cost filtering applies to all the information exchanged;

It does not require the support of the End point Information Services because an uCDN does not request individual endpoints information to a dCDN.

The Sessions Handler maintains the logical association between an uCDN and a dCDN. It controls the session according to the session parameters: It handles the filtering of the network Map and of the Cost Map according the PoINTs and of the CoINTs of the session.

The Sessions Handler handles the views given by the configuration of the session.

Information Services are accessible through HTTP GET messages only.

A dCDN ALTO server does not expose the URIs nor provides an Information Resource Directory.

The Map Filtering logic and the sessions handler replace the Map Filtering service of the current version of the ALTO protocol.

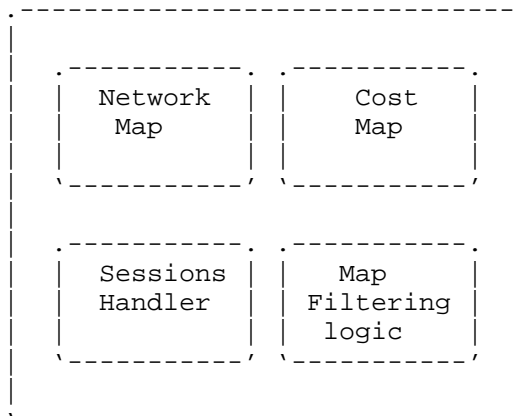


Figure 5: ALTO Protocol for CDN interconnection

4.2. View Configuration

Views are similar to pre-generated maps presented in the section 7.6.3. of [I-D.ietf-alto-protocol]. Their configurations are local to the ALTO server.

The configuration includes a name, a PoINT and a CoINT. A view provides an ALTO CLient with at least 2 Information Resources: the network map associated with the PoINT and the cost map associated with the CoINT.

Its definition includes the setting of the URIs towards these pre-generated maps.

N.B.: Cost of Interest (CoINT) will be defined in the next version of the document.

4.2.1. PoINT

A PoINT applies at the view level. It specifies a local filter tied to an URI which provides the ALTO client with a link to download the output of this filter (see examples in section 4.2.2). It applies during all the duration of the session.

This filter produces pre-generated maps. The output of the filter is a pre-generated network map and a optional pre-generated Network Map Status. The Network Map Status will be specified in the next version of the document.

Note: Filters value are not exchanged over the network. Nevertheless, as ALTO Maps are JSON documents it is desirable to write the filter using JSON document filtering mechanisms like JSON pointer [I-D.pbryan-zyp-json-pointer]. In this memo the filters are defined using a syntax similar to JSON pointer but allowing wildcard (like W3C XPATH does).

4.2.2. View Configuration Examples

Following are the PoINT corresponding to the use case of the section 2.

Note: View Configurations are internal to an ALTO Server. They are not exchanged between the ALTO server and the ALTO Client. In this section the View Configurations are written in JSON to ease the reading only.

```

{
  "view" : "ipv4",
  "point" : {
    "filter": "map/PID_*/ipv4" ,
    "map" : "http://cdni.alto.example.com/CDN1/networkmap/ipv4"
    "map_status" : "http://cdni.alto.example.com/CDN1/networkmap/ipv4/status"
  }
  "coint" : []
}
CDN1 IPv4 view

{
  "view" : "FTTH",
  "point" : {
    "filter": "map/PID_FTTH" ,
    "map" : "http://cdni.alto.example.com/CDN2/networkmap/FTTH"
    "map_status" : "http://cdni.alto.example.com/CDN2/networkmap/FTTH/status"
  }
  "coint" : []
}
CDN2 FTTH view

{
  "view" : "IPv6",
  "point" : {
    "filter": "map/PID_*/IPv6" ,
    "map" : "http://cdni.alto.example.com/CDN2/networkmap/IPv6"
    "map_status" : "http://cdni.alto.example.com/CDN2/networkmap/ipv6/status"
  }
  "coint" : []
}
CDN2 IPv6 view

```

4.3. Session Configuration Parameters

The agreement between uCDN and dCDN operators defines the configuration set of the ALTO session.

The configuration of the ALTO interface between an uCDN and a dCDN requires the exchange of session parameters between the two CDNs operators. This can be performed either out-of-band or through the Cdni Control interface. In both cases the setting of a Cdni ALTO session requires an agreement between the 2 CDNs operators and a technical description of the session configuration (addresses, URL, authentication methods, etc.), of the information which can be exchanged (PID filtering, level of details of the maps) and of the way the information is exchanged (update procedure, time scale of updates).

The session configuration relies on the following parameters:

connection: server and client addresses, URL base, authentication methods, etc.;

session_filter: The PIDs which are in the scope of the session. The Cost parameters which are in the scope of the session;

views: a list of views;

POINTs;

CoINTs;

nmap_heartbeat: Order of magnitude of the periodicity of the download of the Network Maps by the client (e.g. every minute or every week);

cmap_heartbeat: Order of magnitude of the periodicity of the download of the Cost Maps by the client (e.g. every minute or every week);

4.4. Error Handling

Errors are reported using legacy ALTO and HTTP errors.

5. Enhancements

The ALTO session presented in this memo relies on basic ALTO services to permit a good level of interoperability, performance and security and to help preserving individual CDN know-how. This section discussed enhancements to a Cdni ALTO session.

5.1. Incremental Update

There are different ways to implement maps update [I-D.schwan-alto-incr-updates]. Two important aspects are to be considered: Improving the processing time in the ALTO client and providing auCDN with relevant updates. An update based on the diff of JSON file entries is useful but not optimized with regard to the uCDN processing because it requires the re-processing of the whole map from scratch after each upload. A better approach consists in defining an update mechanism providing the diff for a grouping of entries such as PIDs.

The approach proposed in section 4.2.1 does not optimize the protocol but provide additionnal Information Resources to let the uCDN

optimizes it exchanges according to its logic.

5.1.1. Level of Details of a Map

The level of information exchanged between a dCDN ALTO server and a uCDN ALTO client must be customizable in order to decrease the amount of exchanged data while providing the required information.

uCDN may not need the full details of each entry map. It may need the details later.

Furthermore there are cases where an uCDN needs only the list of the PIDs of dCDN. This covers situations where the very detail of each PID of a Network Map is available over existing interfaces like BGP.

For these reasons an uCDN ALTO client should be allowed to get only the summary of the maps (e.g. the list of the PIDs of a Network Map). This can be achieved by defining additional session parameters which set the level of detail of the maps.

5.2. Bi-directional Exchange of Information

In the Cdni context, tied interactions between an uCDN and a dCDN may be required. As Discussed in section 3, there are different aspects requiring a Bi-directional exchange of information including:

Network Map Update Notifications: The update mechanism based on HTTP download is sub-optimal when the uCDN requires a real time propagation of the updates. Bi-directional interactions allow adCDN to notify auCDN with Network Map updates;

Exposition of uCDN constraints: Allowing an uCDN to inform dCDN about its high level constraints like forecast indications provides dCDN with valuable information for optimizing its resources provisioning;

Session Customization: There are situations where an uCDN may require other Views or modify existing Views and where there is a high level of trust between the two CDNs. Consequently the ALTO session might support the modification of the Views by the auCDN.

One solution consists in upgrading the HTTP session to a bi directional protocol. WebSocket Protocol [RFC6455] is one potential candidate.

6. IANA Considerations

This document will request the registration of the media type corresponding to new information services introduced, if any.

7. Security Considerations

This memo defines an ALTO session for CDN interconnection. It specifies a mean to manage finely the information exchanged over the ALTO protocol. By reducing the information exposed it increase the security in general.

Performance:

The usage of the ALTO services by the client may stress the server. Consequently the volume and the number of these messages may affect the availability and the performance of the ALTO server.

Despite the information services provide an uCDN ALTO client with means to control the amount of information downloaded from a dCDN ALTO server it should protect itself from the download of huge network map.

Privacy:

The extension has less privacy concerns than the current ALTO specification because it does not require the support of the End point Information Services.

Know-how protection: unlike section 8 of [I-D.ietf-alto-protocol], an ALTO client is not allowed to redistribute information received from a ALTO server.

8. Acknowledgments

Part of this work is funded by the EU FP7 Envision project.

The authors would like to thank Christian Jacquenet for its feedbacks on preliminary versions of this document.

9. References

9.1. Normative References

[I-D.ietf-alto-protocol]

Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol", draft-ietf-alto-protocol-11 (work in progress), March 2012.

[I-D.ietf-cdni-requirements]

Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", draft-ietf-cdni-requirements-03 (work in progress), June 2012.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

[I-D.ietf-alto-deployments]

Stiemerling, M., Kiesel, S., and S. Previdi, "ALTO Deployment Considerations", draft-ietf-alto-deployments-04 (work in progress), March 2012.

[I-D.ietf-alto-server-discovery]

Kiesel, S., Stiemerling, M., Schwan, N., Scharf, M., and S. Yongchao, "ALTO Server Discovery", draft-ietf-alto-server-discovery-03 (work in progress), March 2012.

[I-D.ietf-cdni-problem-statement]

Niven-Jenkins, B., Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", draft-ietf-cdni-problem-statement-08 (work in progress), June 2012.

[I-D.ietf-cdni-use-cases]

Bertrand, G., Emile, S., Burbridge, T., Eardley, P., Ma, K., and G. Watson, "Use Cases for Content Delivery Network Interconnection", draft-ietf-cdni-use-cases-08 (work in progress), June 2012.

[I-D.jenkins-alto-cdn-use-cases]

Niven-Jenkins, B., Watson, G., Bitar, N., Medved, J., and S. Previdi, "Use Cases for ALTO within CDNs", draft-jenkins-alto-cdn-use-cases-03 (work in progress), June 2012.

[I-D.marocco-alto-next]

Marocco, E. and V. Gurbani, "Extending the Application-Layer Traffic Optimization (ALTO) Protocol", draft-marocco-alto-next-00 (work in progress), January 2012.

[I-D.medved-alto-svr-apis]

Medved, J., Ward, D., Peterson, J., Woundy, R., and D. McDysan, "ALTO Network-Server and Server-Server APIs", draft-medved-alto-svr-apis-00 (work in progress), March 2011.

[I-D.pbryan-json-patch]

Bryan, P., "JSON Patch", draft-pbryan-json-patch-04 (work in progress), December 2011.

[I-D.pbryan-zyp-json-pointer]

Bryan, P. and K. Zyp, "JSON Pointer", draft-pbryan-zyp-json-pointer-02 (work in progress), October 2011.

[I-D.penno-alto-cdn]

Penno, R., Medved, J., Alimi, R., Yang, R., and S. Previdi, "ALTO and Content Delivery Networks", draft-penno-alto-cdn-03 (work in progress), March 2011.

[I-D.previdi-cdni-footprint-advertisement]

Previdi, S., Faucheur, F., Faucheur, F., Medved, J., and L. Faucheur, "CDNI Footprint Advertisement", draft-previdi-cdni-footprint-advertisement-01 (work in progress), March 2012.

[I-D.randriamasy-alto-multi-cost]

Randriamasy, S. and N. Schwan, "Multi-Cost ALTO", draft-randriamasy-alto-multi-cost-06 (work in progress), March 2012.

[I-D.schwan-alto-incr-updates]

Schwan, N. and B. Roome, "ALTO Incremental Updates", draft-schwan-alto-incr-updates-01 (work in progress), March 2012.

[RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, October 2009.

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011.

Authors' Addresses

Emile Stephan
France Telecom - Orange
2 avenue Pierre Marzin
Lannion F-22307
France

Email: emile.stephan@orange.com

Selim Ellouze
France Telecom - Orange
2 avenue Pierre Marzin
Lannion F-22307
France

Email: selim.ellouze@orange.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 19, 2014

E. Stephan
Orange
S. Ellouze
H-log
April 17, 2014

ALTO session for CDN Interconnection
draft-stephan-cdni-alto-session-ext-05

Abstract

The selection of a downstream CDN by an upstream CDN is based on multi-dimensional criteria. Various protocols, such as BGP or ALTO, may be used by a downstream CDN to expose content routing information and interconnection preferences to an upstream CDN. The selection of such a protocol is premature as the WG, and especially the Footprint/Capabilities Design Team, is currently working on this topic. So this draft does not promote the usage of the ALTO protocol for CDN interconnection. It presents the limitations of the current ALTO protocol in the case it would be selected for CDN interconnection. It specifies the mechanism for controlling the session initialization and for limiting the information exchanged. Then it discusses the need of incremental update and proposes to study the usage of Netconf /Yang to provide ALTO server with notifications.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 19, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
1.1. Terminology	4
2. Use Cases	5
2.1. CDN1 views	8
2.2. CDN2 views	8
2.3. Map Maintenance	9
3. Requirements for an ALTO Session for Cdni	9
3.1. ALTO Information Customization	9
3.2. View download with HTTP GET	10
3.3. Initialization of the Session	10
3.4. Server Discovery	10
3.5. Asynchronous Maps Update	11
3.6. Information Resource Directory	11
3.7. PID Stability	11
3.8. Scalability	12
3.9. Performance	12
3.10. dCDN Traffic Optimization	12
4. Specification of the ALTO Session for Cdni	13
4.1. Cdni ALTO session Framework	13

4.2.	View Configuration	14
4.2.1.	POINT	14
4.2.2.	View Configuration Examples	15
4.3.	Session Configuration Parameters	16
4.4.	Error Handling	16
5.	Expected Enhancements	16
5.1.	Asynchronous Updates	16
5.2.	Incremental Download of the Updates	16
5.2.1.	Level of Details of a Map	17
5.3.	Bi-directional Exchange of Information	17
6.	Extension for Asynchronous update	17
7.	IANA Considerations	18
8.	Security Considerations	18
9.	Acknowledgments	18
10.	References	18
10.1.	Normative References	18
10.2.	Informative References	19
	Authors' Addresses	21

1. Introduction

The selection of a downstream CDN by an upstream CDN is based on multi-dimensional criteria. Various protocols, such as BGP or ALTO, may be used by a downstream CDN to expose content routing information and interconnection preferences to an upstream CDN. The selection of such a protocol is premature as the WG, and especially the Footprint/Capabilities Design Team, is currently working on this topic. So this draft does not promote the usage of the ALTO protocol for CDN interconnection. It presents the limitations of the current ALTO protocol in the case it would be selected for CDN interconnection. It specifies the mechanism for controlling the session initialization and for limiting the information exchanged. Then it discusses the need of incremental update and proposes to study the usage of Netconf /Yang to provide ALTO server with notifications.

Currently the ALTO protocol is designed for the communication of network information to untrusted internet applications. In the context of a CDN interconnection (CDNi) there is a certain level of trust, at least enough to mount a subset of the interfaces depicted in [RFC6707]. In practice the level of trust differs with each interconnection. There are situations where a CDNi ALTO server has to exchange information with an ALTO client of an affiliate and with an ALTO client of a competitor (see [RFC6770]).

In the first case topology hiding [RFC5693] may not be required. In the second case the operator of a dCDN may consider a fine control of the exposed information. Consequently the ALTO server of a dCDN operator must be able to adapt the information exposed to each uCDN.

The document discusses firstly the insightful aspects of such a use cases in section 2. Then in section 3 it presents the motivations for specifying an ALTO session to customize the information exposed in each CDN interconnection. In section 4, it provides a proposal for an appropriate specification of an ALTO session for a CDN interconnection. Then in section 5 it discusses different enhancements of interest to a Cdni ALTO session. Finally in the section it studies the usage of Netconf/Yang works to provide ALTO with server notifications.

N.B.: this version of the memo covers only the Network Map.

1.1. Terminology

The reader must be familiar with the terminology given by the drafts [RFC6707], and [I-D.ietf-cdni-requirements] , and [I-D.ietf-alto-protocol].

The following abbreviations are recalled:

dCDN : downstream CDN: The CDN which provide the delivery resource;

uCDN : upstream CDN: The CDN which may rely on dCDN server to deliver contents;

PID : Provider-defined Network Location Identifier;

NSP : Network Service Provider (e.g. ISP connecting End User to Internet);

ALTO Information Resource Directory: (Directory): The Information Resource Directory indicates to ALTO Clients which Information Resources are made available by an ALTO Server (section 7.6 [I-D.ietf-alto-protocol]).

Following are terms and abbreviations introduced in the document:

adCDN: ALTO downstream CDN server;

auCDN: ALTO upstream CDN client;

PIDs of Interest (PoINT) : The PIDs which are in the scope of an ALTO session or of a view. They may be defined as a list or by a XSLT-like statement (e.g. 'map/*/ipv6');

Costs of Interest (CoINT) : The Costs which are in the scope of an ALTO session or of a view;

ALTO Client-Server session: The logical association between an Cdni ALTO Client and an Cdni ALTO server which maintains the context across the ALTO HTTP connections made by the client to the server.

View: A view is the set of URIs which provide an auCDN with a mean for downloading the maps reflecting an agreement between an uCDN and a dCDN. A view is defined by PIDs of Interest (PoINT) and Costs of Interest (CoINT).

2. Use Cases

This section depicts a situation where a dCDN exposes information according to the agreements of each CDN interconnection. The information is exposed within an ALTO session based on the current ALTO protocol version. There is not time dependency between the content requests received by the upstream CDN and the information exchanged over the Cdni ALTO interface.

To ease the reading, the content of the Network Maps is intentionally limited with regards to real situation.

The use case is about a NSP which deployed a CDN named CDN0 over its network and where CDN0 acts as a downstream CDN for two CDNs named CDN1 and CDN2. CDN1 is an affiliate of CDN0.

In the figure 1, the network of NSP provides CDN0 with an aggregated view of the routing information. The grouping of the routing information results from the processing of information provided by BGP according to various policies of the NSP (network, content distribution, etc).

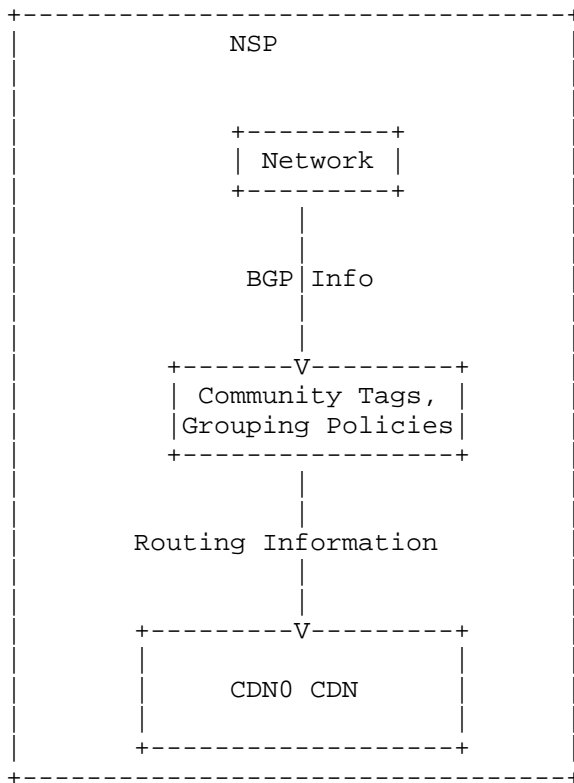


Figure 1: Internal Routing Information

The Figure 2 shows CDN0 acting as a dCDN for CDN1 and CDN2. CDN0 ALTO server (adCDN0) filters and sends stable Network and Cost Maps to the uCDN ALTO clients according to its policies and with respect to the peering agreement between the NSP and the operators of CDN1 and CDN2. adCDN0 is connected to 2 Cdni ALTO clients named auCDN1 and auCDN2.

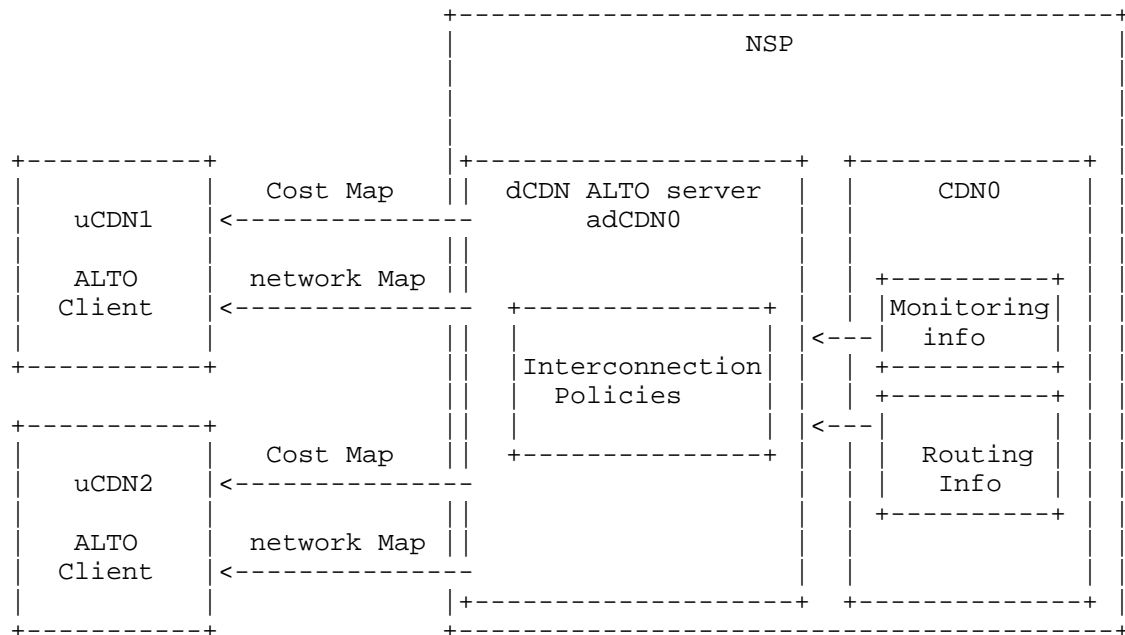


Figure 2: CDNs interconnection

The Figure 3 presents the internal representation of the Network Map computed by CDN0 ALTO server.

```

"map" : {
  "PID_DSL" : {
    "ipv4" : [
      "192.0.2.0/24",
      "198.51.100.0/25"
    ],
    "ipv6" : [
      "2001:db8:0:1::/64",
    ]
  },
  "PID_FTTH" : {
    "ipv4" : [
      "198.51.100.128/25"
    ],
    "ipv6" : [
      "2001:db8:0:2::/64"
    ]
  }
}

```


Figure 3: CDN0 internal Network Map

2.1. CDN1 views

CDN0 and CDN1 agreed that CDN1 needs only the IPv4 view of the Network Map. The Network Map, presented in figure 4, is downloadable by CDN1 at the URI 'http://cdni.alto.example.com/CDN1/networkmap/ipv4'.

```
"map" : {
  "PID_DSL" : {
    "ipv4" : [
      "192.0.2.0/24",
      "198.51.100.0/25"
    ]
  },
  "PID_FTTH" : {
    "ipv4" : [
      "198.51.100.128/25"
    ]
  }
}
```

Figure 4: CDN1 IPv4 Network Map view

2.2. CDN2 views

CDN0 and CDN2 have 2 separate agreements. Both are relative to the geographical extension of CDN2 coverage. The first agreement concerns the exposition of FTTH customers only. The second one covers IPv6 customers only. They are reflected as separated Network Maps. The first Network Map, exposed in figure 5, is downloadable by CDN2 at the URI 'http://cdni.alto.example.com/CDN2/networkmap/FTTH'.

```
"map" : {
  "PID_FTTH" : {
    "ipv4" : [
      "198.51.100.128/25"
    ],
    "ipv6" : [
      "2001:db8:0:2::/64"
    ]
  }
}
```

Figure 5: CDN2 FTTH Network Map.

The second Network Map, exposed in the figure 6, is downloadable by auCDN2 at the URI 'http://cdni.alto.example.com/CDN2/networkmap/IPv6'.

```
"map" : {  
  "PID_DSL" : {  
    "ipv6": [  
      "2001:db8:0:1::/64",  
    ]  
  },  
  "PID_FTTH" : {  
    "ipv6": [  
      "2001:db8:0:2::/64"  
    ]  
  }  
}
```

Figure 6: CDN2 IPv6 Network Map

2.3. Map Maintenance

auCDN1 and auCDN2 need a mean for maintaining the content of the maps. The ALTO server of CDN0 provides each view with an URI towards a list of the PIDs which were really modified in the last update. Each dCDN can download this information and determine whenever or not it have to update the Network Map of this view again.

This is not optimal. Nevertheless it provides an update mechanism based on HTTP GET which contribute to the reduction of both the volume of information exchanged and the processing on each side.

3. Requirements for an ALTO Session for Cdni

This section motivates the necessity of specifying an ALTO session between a dCDN and a uCDN with adequate features for addressing CDN interconnection requirements.

3.1. ALTO Information Customization

The current ALTO approach excludes that the Maps exposed by the ALTO server differ according to the client. This is enforced by section 7.2.6 of [I-D.ietf-alto-protocol] which recommends ignoring HTTP session parameters and HTTP cookies.

CDNi widely differs in such aspects because a dCDN operator must be able to adapt the information exposed to each uCDN according first to its policies and second to its agreements with uCDN. Moreover it is important for a uCDN client to optimize the volume and the relevance

of the information received by avoiding downloading unwanted information in order to enhance the performance of the processing of the received data.

Consequently the customization of the ALTO interface between an uCDN and a dCDN requires the specification of a minimal set of session parameters. This must be specified inside the Cdni WG to provide a minimal level of interoperability amongst CDNs.

3.2. View download with HTTP GET

Currently [I-D.ietf-alto-protocol] (section 7.6.2 and 7.6.4.1) allows two Information Resources of the Information Resource Directory to match the same view of a map and to be downloadable using either an HTTP POST or a HTTP GET.

In the context of ALTO session for Cdni this is not allowed. A view of a map is accessible by a unique URI using HTTP GET request only. The session configuration defines all the information resources that an auCDN can download.

3.3. Initialization of the Session

The setting of the session between an uCDN and a dCDN reflects their agreements and expectancies. A minimal configuration of the session is required for ensuring an efficient initialization of the interface, for decreasing the service time, increasing the interoperability and improving the security.

The exchange of the session configuration parameters can be performed either out-of-band (human settings) or through the Cdni Control interface. In both cases the setting of a Cdni ALTO session requires an agreement between the 2 CDNs operators and a technical description of the session configuration (ALTO server and client addresses, URL, authentication methods, etc.), of the information which can be exchanged (PID of Interest, Cost of interest, level of details of the maps, etc) and of the way the information is exchanged (update method, time-scale for updates, etc).

3.4. Server Discovery

The discovery of a dCDN server by a uCDN relies on parameters exchanged out-of-band or on the Cdni Control interface. Consequently a CDN interconnection does not require any discovery mechanisms like described in [I-D.ietf-alto-server-discovery].

3.5. Asynchronous Maps Update

The way the update is implemented is under discussion in the ALTO WG because there is a strong need to optimize the volume of information exchanged during the update and the processing of the information. [I-D.schwan-alto-incr-updates] presents solutions for incremental download where the auCDN download the diff of the Maps.

Incremental download does not resolve the case where auCDN require to be informed in real time each time an information changes. In this case, the adCDN must push the updates on the fly in notifications towards the auCDN.

3.6. Information Resource Directory

Section 7.6 of [I-D.ietf-alto-protocol] requires the availability of Information Resource Directory for exposing the Information Resources (i.e. URIs of the maps).

In a Cdni interconnection it is not necessary to provide such directory as the two interconnected CDNs previously agreed on the URI names. Besides, avoiding the exposition of URIs enhances the security of the system (see section 11.5. [I-D.ietf-alto-protocol]).

3.7. PID Stability

Currently ALTO servers scramble the prefixes among the PIDs to prevent reverse engineering by ALTO clients ([I-D.ietf-alto-protocol], section 12.1).

CDNi situations differ widely in such aspects. Such nondeterministic semantics is totally unusable by a request routing function of a uCDN, or may lead to suboptimal decision. The dCDN must expose meaningful information to uCDN. Consequently the meaning of the PIDs must not change during the session duration.

As described in section 4.1 of [I-D.previdi-cdni-footprint-advertisement] a CDN acquires part of the content routing information from legacy BGP. As given by figure 1, The NSP may use part of the community tags carried by its legacy internal BGP to filter and gather the prefixes in stable groups (see section 5.1.7 of [I-D.ietf-alto-deployments]) that may then be used by its internal CDN [I-D.jenkins-alto-cdn-use-cases].

3.8. Scalability

The routing function of an uCDN might not require all the information that an ALTO server of an dCDN might expose. Furthermore, as per nature an uCDN interconnects with several dCDNs, this volume might harm its performance and its reliability [I-D.ietf-alto-deployments].

The same applies for dCDN ALTO server. It must not be overloaded by uCDNs requests.

Consequently the Cdni ALTO session will provide dCDN and uCDN with a mean to shape the information to exchange in an interoperable manner. For instance, an uCDN may not want to receive the very last detailed level of the network map of all the dCDNs it is interconnected with; it may not want to receive each update; it may be interested only by one cost type attribute of the Cost Map service, etc.

N.B.: The situation will be even worse when the maps will include multi-cost as proposed by ([I-D.randriamasy-alto-multi-cost] and [I-D.marocco-alto-next] section 3.2) because the size of the maps will increase.

3.9. Performance

The amount of information to be processed impacts directly the performance of an auCDN. As an example an uCDN does not want to download all the PIDs when it needs only the PIDs of the Endpoints managed directly by each dCDN. Currently, as given by section 7.2.2. of [I-D.ietf-alto-protocol] , this is achieved using HTTP POST querying the ALTO Map Filtering Service or by HTTP GET of pre-generated maps.

To optimize the performance the ALTO Map Filtering Service is not exposed. Map filtering is accessible only through HTTP GET toward pre-generated maps according to the configuration of the session agreed by the CDNs. Consequently the ALTO session configuration must include must include filters (PIDs, cost, etc) to reduce the volume of information exchanged about to the PIDs of Interest (PoINT) and to the Cost of Interest (CoINT) agreed by uCDN and dCDN operators. These filters apply during all the duration of the ALTO session.

3.10. dCDN Traffic Optimization

Considering that ALTO is about traffic optimization at the application level, in the context of a Cdni interconnection between an uCDN and a dCDN, ALTO is capable of covering the exchange of information from dCDN to uCDN, allowing for the optimization of the delivery at the uCDN side only. In contrast, exchanging information

the other way around for allowing delivery optimization at the dCDN level is not addressed yet.

Indeed a dCDN is subject to rival uCDNs requesting resources based on information exposed by the dCDN. By exposing their constraints and their needs, the uCDNs requirements are better addressed by dCDNs through a smart resource provisioning and sharing.

A uCDN should be able to provide dCDN with information that may help dCDN to optimize its resources.

4. Specification of the ALTO Session for Cdni

This section specifies the ALTO session for Cdni.

4.1. Cdni ALTO session Framework

The figure 7 presents the Framework of the ALTO Session for Cdn interconnection:

The Map filtering logic is represented to reflect the customization of the content of the internal maps to the server according to the scope of the sessions with uCDNs.

There are filters to limit the scope of the session with regard to the content of the internal maps content of the server.

Network Map and Cost Map are unchanged. Nevertheless session filtering applies to all the information exchanged;

It does not require the support of the End point Information Services because an uCDN does not request individual endpoints information to a dCDN.

The Sessions Handler maintains the logical association between an uCDN and a dCDN. It controls the session according to the session parameters: It handles the filtering of the network Map and of the Cost Map according to the PoINTs and of the CoINTs of the session.

The Sessions Handler handles the views given by the configuration of the session.

Information Services are accessible through HTTP GET messages only.

A dCDN ALTO server does not expose the URIs nor provides an Information Resource Directory.

The Map Filtering logic and the sessions handler are similar to the Map Filtering service of the current version of the ALTO protocol.

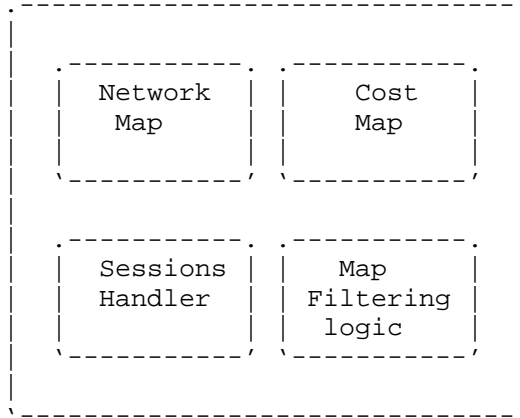


Figure 5: ALTO Protocol for CDN interconnection

4.2. View Configuration

Views are similar to pre-generated maps presented in the section 7.6.3. of [I-D.ietf-alto-protocol]. Their configurations are local to the ALTO server.

The configuration includes a name, a PoINT and a CoINT. A view provides an ALTO CLient with at least 2 Information Resources: the network map associated with the PoINT and the cost map associated with the CoINT.

Its definition includes the setting of the URIs towards these pre-generated maps.

N.B.: Cost of Interest (CoINT) will be defined in a next version of the document.

4.2.1. PoINT

A PoINT applies at the view level. It specifies a local filter tied to an URI which provides the ALTO client with a link to download the output of this filter (see examples in section 4.2.2). It applies during all the duration of the session.

This filter produces pre-generated maps. The output of the filter is a pre-generated network map and an optionnal pre-generated Network Map Status. The Network Map Status will be specified in a future version of the document.

4.2.2. View Configuration Examples

Following are the views corresponding to the use case of the section 2.

```
{
  "view" : "ipv4",
  "point" : {
    "filter": "map/PID_*/ipv4" ,
    "map" : "http://cdni.alto.example.com/CDN1/networkmap/ipv4"
    "map_status" : "http://cdni.alto.example.com/CDN1/networkmap/ipv4/status"
  }
  "coint" : []
}
```

CDN1 IPv4 view

```
{
  "view" : "FTTH",
  "point" : {
    "filter": "map/PID_FTTH" ,
    "map" : "http://cdni.alto.example.com/CDN2/networkmap/FTTH"
    "map_status" : "http://cdni.alto.example.com/CDN2/networkmap/FTTH/status"
  }
  "coint" : []
}
```

CDN2 FTTH view

```
{
  "view" : "IPv6",
  "point" : {
    "filter": "map/PID_*/IPv6" ,
    "map" : "http://cdni.alto.example.com/CDN2/networkmap/IPv6"
    "map_status" : "http://cdni.alto.example.com/CDN2/networkmap/ipv6/status"
  }
  "coint" : []
}
```

CDN2 IPv6 view

4.3. Session Configuration Parameters

The agreement between uCDN and dCDN operators defines the configuration set of the ALTO session. The configuration of the ALTO interface between an uCDN and a dCDN requires the exchange of session parameters between the two CDNs operators. This can be performed either out-of-band (by phone call, etc) or through the Cdni Control interface. In both cases the setting of a Cdni ALTO session requires an agreement between the 2 CDNs operators and a technical description of the session configuration (Server addresses, URL, authentication methods, etc.), of the information which can be exchanged (PID filtering, level of details of the maps) and of the way the information is exchanged (update procedure, etc).

The session configuration relies on the following parameters:

connection: server and client addresses, URL base, authentication methods, etc.;

session_filter: The PIDs which are in the scope of the session. The Cost parameters which are in the scope of the session;

views: a list of views;

4.4. Error Handling

Errors are reported using legacy ALTO and HTTP errors.

5. Expected Enhancements

This section discussed enhancements which might be required to improve a Cdni ALTO session.

5.1. Asynchronous Updates

In the Cdni context, there are tied interactions between an uCDN and a dCDN interconnected. It requires generally a high level of synchronization of the Maps of the dCDN and of the uCDN. The update mechanism based on HTTP download is sub-optimal when the uCDN requires a real time propagation of the updates. To meet this requirement the adCDN must notify the update to adCDN.

5.2. Incremental Download of the Updates

Incremental download reduces the volume of the information exchanged. An update based on the diff of JSON file entries is useful but not optimized because it requires the re-processing of the whole map from scratch after each upload. A better approach might consist in

defining an update mechanism providing the diff for a grouping of entries such as PIDs. T

The drawback is that incremental download does not provide a high level of synchronization of the Maps of the dCDN and of the uCDN.

5.2.1. Level of Details of a Map

The level of information exchanged between a dCDN ALTO server and a uCDN ALTO client must be customizable in order to decrease the amount of exchanged data while providing the required information.

uCDN may not need the full details of each entry map or it may need the details later.

Furthermore there are cases where an uCDN needs only the list of the PIDs of dCDN (e.g. the very detail of each PID of a Network Map is available over existing interfaces like BGP).

For these reasons an uCDN ALTO client should be allowed to get only the summary of the maps (e.g. the list of the PIDs of a Network Map). This can be achieved by defining additional session configuration parameters which set the level of detail of the maps.

5.3. Bi-directional Exchange of Information

As Discussed in section 3, there are different aspects requiring a Bi-directional exchange of information including:

Exposition of uCDN constraints: Allowing an uCDN to inform dCDN about its high level constraints like forecast indications provides dCDN with valuable information for optimizing its resources provisioning;

Session Customization: There are situations where an uCDN may require other Views or modify existing Views and where there is a high level of trust between the two CDNs. Consequently the ALTO session might support the modification of the Views by the auCDN.

6. Extension for Asynchronous update

There are many ways to address the enhancements expected in the section 5.

One solution consists in upgrading the HTTP session to a bi-directional protocol and in specifying an asynchronous update mechanism. Netconf [RFC6241] and YANG [RFC6020] works fill this gap. Netconf already include the specification of notifications [RFC6470]

based on subscriptions [RFC5277]. Maps update information can be inserted into NETCONF notifications or updated as YANG or JSON patch using the method being specified by the NETCONF WG in the draft [I-D.ietf-netconf-yang-patch].

7. IANA Considerations

none.

8. Security Considerations

This memo defines an ALTO session for CDN interconnection. It specifies a mean to manage finely the information exchanged over the ALTO protocol. By reducing the information exposed it increase the security in general.

Performance:

The usage of the ALTO services by the client may stress the server. Consequently the volume and the number of these messages may affect the availability and the performance of the ALTO server.

Despite the information services provide an uCDN ALTO client with means to control the amount of information downloaded from a dCDN ALTO server it should protect itself from the download of huge network map.

Privacy:

The extension has less privacy concerns than the current ALTO specification because it does not require the support of the End point Information Services.

9. Acknowledgments

The authors would like to thank Christian Jacquenet for its feedbacks on preliminary versions of this document.

10. References

10.1. Normative References

[I-D.ietf-alto-protocol]

Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol", draft-ietf-alto-protocol-27 (work in progress), March 2014.

- [I-D.ietf-cdni-requirements]
Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", draft-ietf-cdni-requirements-17 (work in progress), January 2014.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", RFC 6707, September 2012.
- [RFC6770] Bertrand, G., Stephan, E., Burbridge, T., Eardley, P., Ma, K., and G. Watson, "Use Cases for Content Delivery Network Interconnection", RFC 6770, November 2012.

10.2. Informative References

- [I-D.ietf-alto-deployments]
Stiemerling, M., Kiesel, S., Previdi, S., and M. Scharf, "ALTO Deployment Considerations", draft-ietf-alto-deployments-09 (work in progress), February 2014.
- [I-D.ietf-alto-server-discovery]
Kiesel, S., Stiemerling, M., Schwan, N., Scharf, M., and S. Yongchao, "ALTO Server Discovery", draft-ietf-alto-server-discovery-10 (work in progress), September 2013.
- [I-D.ietf-appsawg-json-patch]
Bryan, P. and M. Nottingham, "JSON Patch", draft-ietf-appsawg-json-patch-10 (work in progress), January 2013.
- [I-D.ietf-netconf-yang-patch]
Bierman, A., Bjorklund, M., Watsen, K., and R. Fernando, "YANG Patch Media Type", draft-ietf-netconf-yang-patch-00 (work in progress), March 2014.
- [I-D.jenkins-alto-cdn-use-cases]
Niven-Jenkins, B., Watson, G., Bitar, N., Medved, J., and S. Previdi, "Use Cases for ALTO within CDNs", draft-jenkins-alto-cdn-use-cases-03 (work in progress), June 2012.
- [I-D.marocco-alto-next]
Marocco, E. and V. Gurbani, "Extending the Application-Layer Traffic Optimization (ALTO) Protocol", draft-marocco-alto-next-00 (work in progress), January 2012.

- [I-D.previdi-cdni-footprint-advertisement]
Previdi, S., Faucheur, F., Faucheur, F., Medved, J., and L. Faucheur, "CDNI Footprint Advertisement", draft-previdi-cdni-footprint-advertisement-02 (work in progress), September 2012.
- [I-D.randriamasy-alto-multi-cost]
Randriamasy, S., Roome, B., and N. Schwan, "Multi-Cost ALTO", draft-randriamasy-alto-multi-cost-07 (work in progress), October 2012.
- [I-D.schwan-alto-incr-updates]
Schwan, N. and B. Roome, "ALTO Incremental Updates", draft-schwan-alto-incr-updates-02 (work in progress), July 2012.
- [NETCONF_YANG_TUT]
"Network Conguration Management with NETCONF and YANG", <<http://cnds.eecs.jacobs-university.de/slides/2012-ietf-84-netconf-yang.pdf>>.
- [RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, July 2008.
- [RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, October 2009.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC6244] Shafer, P., "An Architecture for Network Management Using NETCONF and YANG", RFC 6244, June 2011.
- [RFC6470] Bierman, A., "Network Configuration Protocol (NETCONF) Base Notifications", RFC 6470, February 2012.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

Authors' Addresses

Emile Stephan
Orange
2 avenue Pierre Marzin
Lannion F-22307
France

Email: emile.stephan@orange.com

Selim Ellouze
H-log
5 rue Guy Moquet
Orsay F-91400
France

Email: selim.ellouze@h-log.fr