

DECADE
Internet-Draft
Intended status: Informational
Expires: April 28, 2011

R. Alimi
Google
Y. Yang
Yale University
A. Rahman
InterDigital Communications, LLC
D. Kutscher
NEC
L. Chen
H. Liu
Yale University
October 25, 2010

DECADE Architecture
draft-alimi-decade-arch-01

Abstract

Peer-to-peer (P2P) applications have become widely used on the Internet today and make up a large portion of the traffic in many networks. One technique to improve the network efficiency of P2P applications is to introduce storage capabilities within the network. The DECADE Working Group has been formed with the goal of developing an architecture to provide this capability. This document presents an architecture, discusses the underlying principles and identifies core components and protocols supporting the architecture.

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 April 28, 2011.

Copyright Notice

Copyright (c) 2010 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 BSD License.

Table of Contents

1. Introduction	4
2. Entities	5
2.1. DECADE Storage Servers	5
2.2. DECADE Storage Provider	5
2.3. DECADE Content Providers	5
2.4. DECADE Content Consumers	5
2.5. Content Distribution Application	5
2.6. Application End-Point	6
3. Architectural Principles	6
3.1. Decoupled Control and Data Planes	6
3.2. Immutable Data Objects	7
3.3. Data Object Identifiers	8
3.4. Explicit Control	8
3.5. Resource and Data Access Control through User Delegation	9
3.5.1. Resource Allocation	9
3.5.2. User Delegations	9
4. System Components	10
4.1. Content Distribution Application	12
4.1.1. Data Sequencing and Naming	12
4.1.2. Native Protocols	12
4.1.3. DECADE Client	13
4.2. DECADE Server	13
4.2.1. Access Control	13
4.2.2. Resource Scheduling	14
4.2.3. Data Store	14
4.3. Protocols	14
4.3.1. DECADE Resource Protocol	14
4.3.2. Standard Data Transports	15
4.4. DECADE Data Sequencing and Naming	15
4.5. In-Network Storage Components Mapped to DECADE Architecture	16
4.5.1. Data Access Interface	16
4.5.2. Data Management Operations	16
4.5.3. Data Search Capability	16
4.5.4. Access Control Authorization	16
4.5.5. Resource Control Interface	16
4.5.6. Discovery Mechanism	16
4.5.7. Storage Mode	17
5. Security Considerations	17
6. IANA Considerations	17
7. Informative References	17
Authors' Addresses	17

1. Introduction

Peer-to-peer (P2P) applications have become widely used on the Internet today to distribute contents, and they contribute a large portion of the traffic in many networks. The DECADE Working Group has been formed with the goal of developing an architecture to introduce in-network storage to be used by such applications, to achieve more efficient content distribution. Specifically, in many subscriber networks, it is typically more expensive to upgrade network equipment in the "last-mile", because it can involve replacing equipment and upgrading wiring at individual homes, businesses, and devices such as DSLAMs and CMTSSs. Thus, it can be cheaper to upgrade core infrastructure involving fewer components that are shared by many subscribers. See [I-D.ietf-decade-problem-statement] for a more complete discussion of the problem domain and general discussion of the capabilities to be provided by DECADE.

This document presents a potential architecture of providing in-network storage that can be integrated into content distribution applications. The primary focus is P2P-based content distribution, but the architecture may be useful to other applications with similar characteristics and requirements. In particular, content distribution applications that may split data into smaller pieces for distribution may be able to utilize DECADE.

The design philosophy of the DECADE architecture is to provide only the core functionality that is needed for applications to make use of in-network storage. With such core functionality, the protocol may be simple and easier to support by storage providers. If more complex functionality is needed by a certain application or class of applications, it may be layered on top of the DECADE protocol.

The DECADE protocol will leverage existing transport and application layer protocols and will be designed to work with a small set of alternative IETF protocols.

This document proceeds in two steps. First, it details the core architectural principles that can guide the DECADE design. Next, given these core principles, this document presents the core components of the DECADE architecture and identifies usage of existing protocols and where there is a need for new protocol development.

This document will be updated to track the progress of the DECADE survey [I-D.ietf-decade-survey] and requirements [I-D.gu-decade-reqs] drafts.

2. Entities

2.1. DECADE Storage Servers

DECADE storage servers are operated by DECADE storage providers and provide the DECADE functionality as specified in this memo, including mechanisms to store, retrieve and manage data. A storage provider may typically operate multiple storage servers.

2.2. DECADE Storage Provider

A DECADE in-storage provider deploys and/or manages DECADE servers within a network. A storage provider may also own or manage the network in which the DECADE servers are deployed.

A DECADE storage provider, possibly in cooperation with one or more network providers, determines deployment locations for DECADE servers and determines the available resources for each.

2.3. DECADE Content Providers

DECADE content providers access DECADE storage servers (by way of a DECADE client) to upload and manage data. A content provider can access one or more storage servers. A content provider may be a single process or a distributed application (e.g., in a P2P scenario).

2.4. DECADE Content Consumers

DECADE content consumers access storage servers (by way of a DECADE client) to download data that has previously been stored by a content provider. A content consumer can access one or more storage servers. A content consumer may be a single process or a distributed application (e.g., in a P2P scenario). An instance of a distributed application, such as a P2P application, may both provide content to and consume content from DECADE storage servers.

2.5. Content Distribution Application

A content distribution application is a distributed application designed for dissemination of possibly-large data to multiple consumers. Content Distribution Applications typically divide content into smaller immutable blocks for dissemination.

The term Application Developer refers to the developer of a particular Content Distribution Application.

2.6. Application End-Point

An Application End-Point is an instance of a Content Distribution Application that makes use of DECADE server(s). A particular Application End-Point may be a DECADE Content Provider, a DECADE Content Consumer, or both.

An Application End-Point need not be an active member of a "swarm" to interact with the DECADE storage system. That is, an End-Point may interact with the DECADE storage servers as an offline activity.

3. Architectural Principles

We identify the following key principles.

3.1. Decoupled Control and Data Planes

The DECADE infrastructure is intended to support multiple content distribution applications. A complete content distribution application implements a set of control functions including content search, indexing and collection, access control, ad insertion, replication, request routing, and QoS scheduling. Different content distribution applications can have unique considerations designing the control and signaling functions. For example, a major competitive advantage of many successful P2P systems is their substantial expertise in how to most efficiently utilize peer and infrastructural resources. Many live P2P systems have their specific algorithms in selecting the peers that behave as the more stable, higher-bandwidth sources. They continue to fine-tune such algorithms. In other words, in-network storage should export basic mechanisms and allow as much flexibility as possible to the control planes to implement specific policies. This conforms to the end-to-end systems principle and allows innovation and satisfaction of specific business goals.

Specifically, in the DECADE architecture, the control plane focuses on the application-specific, complex, and/or processing intensive functions while the data plane provides storage and data transport functions.

- o Control plane: Signals details of where the data is to be downloaded from. Also signals the time, quality of service, and receiver of the download. It also provides higher layer meta-data management functions such as defining the sequence of data blocks forming a higher layer content object. These are behaviors designed and implemented by the Application. By Application, we mean the broad sense that include other control plane protocols.

- o Data plane: Stores and transfers data as instructed by the Application's Control Plane.

Decoupling control plane and data plane is not new. For example, OpenFlow is an implementation of this principle for Internet routing, where the computation of the forwarding table and the application of the forwarding table are separated. Google File System applies the principle to file system design, by utilizing the Master to handle the meta-data management, and the chunk servers to handle the data plane (i.e., read and write of chunks of data). NFS4 also implements this principle.

Note that applications may have different Data Plane implementations in order to support particular requirements (e.g., low latency). In order to provide interoperability, the DECADE architecture does not intend to enable arbitrary data transport protocols. However, the architecture may allow for multiple data transport protocols to be used.

Also note that although an application's existing control plane functions remain implemented within the application, the particular implementation may need to be adjusted to support DECADE.

3.2. Immutable Data Objects

A property of bulk contents to be distributed is that they typically are immutable -- once a piece of content is generated, it is typically not modified. It is not common that bulk contents such as video frames and images need to be modified after distribution.

Many content distribution applications divide content objects into blocks for two reasons: (1) multipath: different blocks may be fetched from different content sources in parallel, and (2) faster recovery and verification: individual blocks may be recovered and verified. Typically, applications use a block size larger than a single packet in order to reduce control overhead.

Common applications whose content matches this model include P2P streaming (live and video-on-demand) and P2P file-sharing content. However, other types of applications may additionally match this model.

DECADE adopts a design in which immutable data objects may be stored at a storage server. Applications may consider existing blocks as DECADE data objects, or they may adjust block sizes before storing in a DECADE server.

Focusing on immutable data blocks in the data plane can substantially

simplify the data plane design, since consistency requirements can be relaxed. It also allows effective reuse of data blocks and de-duplication of redundant data.

Depending on specific application requirements, data objects can be complete self-contained resources (such as video files) or chunks of such resources. The DECADE architecture and protocols are agnostic to the nature of the data objects and do not specify a fixed size for them.

Note that immutable content may still be deleted. Also note that immutable data blocks do not imply that contents cannot be modified. For example, a meta-data management function of the control plane may associate a name with a sequence of immutable blocks. If one of the blocks is modified, the meta-data management function changes the mapping of the name to a new sequence of immutable blocks.

3.3. Data Object Identifiers

Objects that are stored in a DECADE storage server can be accessed by DECADE content consumers by a resource identifier that has been assigned within a certain application context.

Because a DECADE content consumer can access more than one storage server within a single application context, a data object that is replicated across different storage servers managed by a DECADE storage provider, can be accessed by a single identifier.

Note that since data objects are immutable, it is possible to support persistent identifiers for data objects.

3.4. Explicit Control

To support the functions of an application's control plane, applications must be able to know and control which data is stored at particular locations. Thus, in contrast with content caches, applications are given explicit control over the placement (selection of a DECADE server), deletion (or expiration policy), and access control for stored data.

Consider deletion/expiration policy as a simple example. Applications may require a DECADE server to store content for a relatively short period of time (e.g. for live-streaming data) or may need to store content long term (e.g., for video-on-demand).

3.5. Resource and Data Access Control through User Delegation

DECADE provides a shared infrastructure to be used by multiple tenants of multiple content distribution applications. Thus, it needs to provide both resource and data access control.

3.5.1. Resource Allocation

There are two primary interacting entities in the DECADE architecture. First, Storage Providers control where DECADE storage servers are provisioned and their total available resources. Second, Applications control data transfers amongst available DECADE servers and between DECADE servers and end-points. A form of isolation is required to enable concurrently-running Applications to each explicitly manage their own content and share of resources at the available servers.

Management of the resources at a server are delegated by a Storage Provider to one or more applications. Applications are able to explicitly and independently manage their own share of resources.

3.5.2. User Delegations

Storage providers have the ability to explicitly manage the entities allowed to utilize the resources at a DECADE server. This capability is needed for reasons such as capacity-planning and legal considerations in certain deployment scenarios.

To provide a scalable way to manage applications granted resources at a DECADE server, a layer of indirection is added. Instead of granting resources to an application, the DECADE server grants a share of the resources to a user. The user may in turn share the granted resources amongst multiple applications. The share of resources granted by a storage provider is called a User Delegation.

A User Delegation may be granted to an end-user (e.g., an ISP subscriber), a Content Provider, or an Application Provider. A particular instance of an application may make use of the storage resources:

- o granted to the end-user (with the end-user's permission),
- o granted to the Content Provider (with the Content Provider's permission>, and/or
- o granted to the Application Provider.

4. System Components

The current version of the document has primarily focused on the architectural principles. The detailed system components will be discussed in the next document revision.

This section presents an overview of the components in the DECADE architecture.

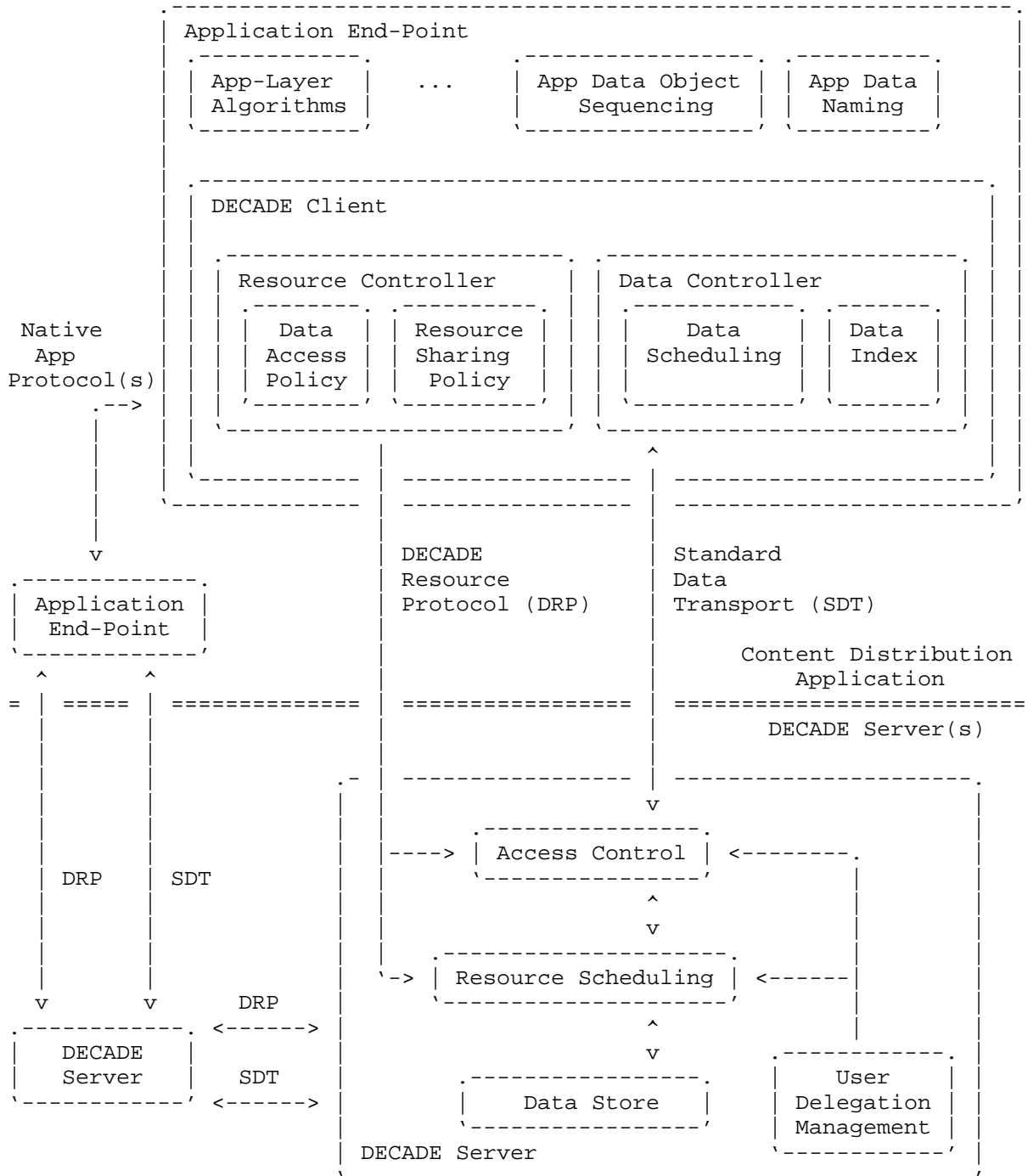


Figure 1: DECADE Architecture Components

A component diagram of the DECADE architecture is displayed in Figure 1. The diagram illustrates the major components of a Content Distribution Application related to DECADE, as well as the functional components of a DECADE Server.

To keep the scope narrow, we only discuss the primary components related to protocol development. Particular deployments may require additional components (e.g., monitoring and accounting at a DECADE server), but they are intentionally omitted from the current version of this document.

4.1. Content Distribution Application

Content Distribution Applications have many functional components. For example, many P2P applications have components to manage overlay topology management, piece selection, etc. In supporting DECADE, it may be advantageous to consider DECADE within some of these components. However, in this architecture document, we focus on the components directly employed to support DECADE.

4.1.1. Data Sequencing and Naming

DECADE is primarily designed to support applications that can divide distributed contents into immutable data objects. To accomplish this, applications include a component responsible for re-assembling data objects and also creating the individual data objects. We call this component Application Data Sequencing. The specific implementation is entirely decided by the application.

In assembling or producing the data objects, an important consideration is the naming of these objects. We call the component responsible for assigning and interpreting application-layer names the Application Data Naming component. The specific implementation is entirely decided by the application.

4.1.2. Native Protocols

Applications may still use existing protocols. Existing protocols used primarily for control/signaling needed by the application, but may also serve as a data transport as they do today; it is important that applications still be designed to be robust (e.g., if DECADE servers are unavailable).

4.1.3. DECADE Client

An application may be modified to support DECADE. We call the layer providing the DECADE support to an application the DECADE Client. It is important to note that a DECADE Client need not be embedded into an application. It could be implemented alone, or could be integrated in other entities such as network devices themselves.

4.1.3.1. Resource Controller

Applications may have different Resource sharing policies and Data access policies to control their resource and data in DECADE servers. These policies can be existing policies of applications (e.g., tit-for-tat) or custom policies adapted for DECADE. The specific implementation is decided by the application.

4.1.3.2. Data Controller

DECADE is designed to decouple the control and the data transport of applications. Data transport between applications and DECADE servers uses standard data transport protocols. It may need to schedule the data being transferred according to network conditions, available DECADE Servers, and/or available DECADE Server resources. An index indicates data available at remote DECADE servers. The index (or a subset of it) may be advertised to other Application End-Points.

4.2. DECADE Server

DECADE server is an important functional component of DECADE. It stores data from Application End-Points, and provides control and access of those data to Application End-Points. Note that a DECADE server is not necessarily a single physical machine, it could also be implemented as a cluster of machines.

4.2.1. Access Control

An Application End-Point can access its own data or other Application End-Point's data (provided sufficient authorization) in DECADE servers. Application End-Points may also authorize other End-Points to store data. If an access is authorized by an Application End-Point, the DECADE Server will provide access.

Note that even if an request is authorized, it may still fail to complete due to insufficient resources by either the requesting Application End-Point or the providing Application End-Point.

4.2.2. Resource Scheduling

Applications may apply their existing resource sharing policies or use a custom policy for DECADE. DECADE servers perform resource scheduling according to the resource sharing policies indicated by Application End-Points as well as configured User Delegations.

Access control and resource control are separated in DECADE server. It is possible that an Application End-Point provides only access to its data without any resources. In order to access this data, another Application End-Point may use the granted access along with its own available resources to store or retrieve data from a DECADE Server.

4.2.3. Data Store

Data from applications may be stored into disks and explicitly or automatically (e.g., after a TTL) deleted from disks. It may be possible to perform optimizations in certain cases, such as avoiding writing temporary data (e.g., live streaming) to disk.

4.3. Protocols

The DECADE Architecture uses two protocols. First, the DECADE Resource Protocol is responsible for communicating access control and resource scheduling policies to the DECADE Server. Second, standard data transport protocols (e.g., WebDAV or NFS) are used to transfer data objects to and from a DECADE Server. The DECADE architecture will specify a small number of Standard Data Transport instances.

Decoupling the protocols in this way allows DECADE to both directly utilize existing standard data transports and to evolve independently.

It is also important to note that the two protocols do not need to be separate on the wire. For example, the DECADE Resource Protocol messages may be piggybacked within extension fields provided by certain data transport protocols. However, this document considers them as two separate functional components for clarity.

4.3.1. DECADE Resource Protocol

The DECADE Resource Protocol is responsible for communicating both access control and resource sharing policies to DECADE Servers used for data transport.

The DECADE architecture specification will provide exactly one DECADE Resource Protocol.

4.3.2. Standard Data Transports

Existing data transport protocols are used to read and write data from a DECADE Server. Protocols under consideration are WebDAV and NFS.

4.4. DECADE Data Sequencing and Naming

We have discussed above that an Application may have its own behavior for both sequencing and naming data objects. In order to provide a simple and generic interface, the DECADE Server is only responsible for storing and retrieving individual data objects.

The issue of naming data objects at the DECADE server would benefit from additional feedback. There are multiple options that have been considered:

- o Self-certifying Name: The name of a data object may be a hash of its contents. Advantages of this scheme include simplicity and low probability of naming collisions without requiring any identifiers or namespaces to be allocated. Disadvantages of this scheme include collision in identifiers (with low probability) and introduction of an additional distribution delay due to the necessity of reading the full object to compute its hash before advertising its availability.
- o Application-specified Name: The name of a data object is specified by the application itself. For example, this could be a function of the application's content identifier and index of the data object. To avoid conflicts, identifiers could be assigned to particular applications. An advantage of this scheme is that collisions could be avoided. A disadvantage is that assigning identifiers introduces additional management complexity.
- o Server-specified Name: The name of a data object is specified by the DECADE server upon initially being stored. An advantage of this approach is that naming conflicts can be completely avoided without requiring particular identifiers to be assigned to applications. A disadvantage is that it introduces additional latency between the time when a application may upload a data object and advertise availability of the data object at the DECADE Server.

The current preferred design is to use self-certifying names. However, additional feedback is welcomed.

4.5. In-Network Storage Components Mapped to DECADE Architecture

In this section we evaluate how the basic components of an in-network storage system identified in Section 3 of [I-D.ietf-decade-survey] map into the DECADE architecture.

It is important to note that complex and/or application-specific behavior is delegated to applications instead of tuning the storage system wherever possible.

4.5.1. Data Access Interface

Users can read and write objects of arbitrary size through the DECADE Client's Data Controller, making use of a standard data transport.

4.5.2. Data Management Operations

Users can move or delete previously stored objects via the DECADE Client's Data Controller, making use of a standard data transport.

4.5.3. Data Search Capability

Users can enumerate or search contents of DECADE servers to find objects matching desired criteria through services provided by the Content Distribution Application (e.g., buffer-map exchanges, a DHT, or peer-exchange). In doing so, End-Points may consult their local data index in the DECADE Client's Data Controller.

4.5.4. Access Control Authorization

All methods of access control are supported: public-unrestricted, public-restricted and private. Access Control Policies are generated by a Content Distribution Application and provided to the DECADE Client's Resource Controller. The DECADE Server is responsible for implementing the access control checks.

4.5.5. Resource Control Interface

Users can manage the resources (e.g. bandwidth) on the DECADE server that can be used by other Application End-Points. Resource Sharing Policies are generated by a Content Distribution Application and provided to the DECADE Client's Resource Controller. The DECADE Server is responsible for implementing the resource sharing policies.

4.5.6. Discovery Mechanism

This is outside the scope of the DECADE architecture. However, it is expected that DNS or some other well known protocol will be used for

the users to discover the DECADE servers.

4.5.7. Storage Mode

DECADE Servers provide an object-based storage mode. Immutable data objects may be stored at a DECADE server. Applications may consider existing blocks as DECADE data objects, or they may adjust block sizes before storing in a DECADE server.

5. Security Considerations

This document currently does not contain any security considerations beyond those mentioned in [I-D.ietf-decade-problem-statement].

6. IANA Considerations

This document does not have any IANA considerations.

7. Informative References

[I-D.ietf-decade-problem-statement]

Yongchao, S., Zong, N., Yang, Y., and R. Alimi, "DECoupled Application Data Enroute (DECADE) Problem Statement", draft-ietf-decade-problem-statement-00 (work in progress), August 2010.

[I-D.ietf-decade-survey]

Alimi, R., Rahman, A., and Y. Yang, "A Survey of In-network Storage Systems", draft-ietf-decade-survey-01 (work in progress), October 2010.

[I-D.gu-decade-reqs]

Yingjie, G., Bryan, D., Yang, Y., and R. Alimi, "DECADE Requirements", draft-gu-decade-reqs-05 (work in progress), July 2010.

Authors' Addresses

Richard Alimi
Google

Email: ralimi@google.com

Y. Richard Yang
Yale University

Email: yry@cs.yale.edu

Akbar Rahman
InterDigital Communications, LLC

Email: akbar.rahman@interdigital.com

Dirk Kutscher
NEC

Email: dirk.kutscher@neclab.eu

Lijiang Chen
Yale University

Email: lijiang.chen@yale.edu

Hongqiang Liu
Yale University

Email: hongqiang.liu@yale.edu

DECADE
Internet-Draft
Intended status: Informational
Expires: April 21, 2011

L. Chen
H. Liu
Yale University
October 18, 2010

Leveraging In-network Storage in P2P LiveStreaming
draft-chen-decade-intgr-livestr-exmp-00

Abstract

DECADE is an in-network storage infrastructure under discussions and constructions. It can be integrated into Peer-to-Peer (P2P) applications to achieve more efficient content distributions. This document represents a detailed example of integrating DECADE with a dominating P2P application - P2P livestreaming. This document describes a preliminary framework of DECADE client API, P2P live streaming integration, the environment of the test on this integration and application performance analysis in the test.

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 April 21, 2011.

Copyright Notice

Copyright (c) 2010 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 BSD License.

Table of Contents

1. Introduction	3
2. Concepts	3
2.1. DECADE Server	3
2.2. DECADE Client Plug-in	3
2.3. P2P LiveStreaming Client	4
3. DECADE Client API	4
4. DECADE Integration of P2P LiveStreaming Client	4
4.1. DECADE Client Plug-in	5
4.2. DECADE Integration Architecture	5
4.2.1. Data Access	5
4.2.2. Message Control	6
5. Test Environment and Settings	6
5.1. Test Settings	6
5.2. Platforms and Components	6
5.2.1. EC2 DECADE Server	7
5.2.2. PlanetLab P2P LiveStreaming Client	7
5.2.3. Tracker	7
5.2.4. Source Server	7
5.2.5. Test Controller	8
6. Performance Analysis	8
6.1. Performance Metrics	8
6.2. Result and Analysis	8
7. Security Considerations	9
8. IANA Considerations	9
9. Normative References	9
Authors' Addresses	9

1. Introduction

DECADE is an in-network storage infrastructure under discussions and constructions. It can be integrated into Peer-to-Peer (P2P) applications to achieve more efficient content distributions.

This draft introduces an instance of application integration with DECADE. In our example system, the core component includes DECADE server and DECADE-aware P2P live stream client. DECADE server running at Linux platform is designed to support data reading and writing for DECADE clients. For DECADE client, we employed a P2P live streaming system called P2PLS (P2P Live Streaming). We utilized a preliminary API (Application Programming Interface) set provided by DECADE to enable P2PLS clients to leverage DECADE in their data transmission. In this draft, we introduce the system structure of the DECADE-P2PLS integration system, the main DECADE related message flow, the environment of the test on this integration, and the system performance in the test.

Please note that P2PLS in this draft only represents the usage case of "live streaming" out of a large number of P2P applications, while DECADE itself can support other applications. The API set of DECADE is only an experimental design and implementation. It is not a standard and is still under development.

2. Concepts

2.1. DECADE Server

DECADE Server is a server implemented DECADE protocols, management mechanism and storage strategies. It is an important element to provide DECADE services. In a DECADE server, we have a number of Data Lockers which are virtual accounts and storage space for applications.

2.2. DECADE Client Plug-in

DECADE Client Plug-in is a plug-in component for application clients to use DECADE server. This plug-in component is served as an application-specific interface between a particular application and DECADE servers. It can be a very simple component of basic DECADE access APIs implemented, or a smart component to integrate application-specific control strategies with DECADE APIs.

2.3. P2P LiveStreaming Client

P2P LiveStreaming Client is our self-maintained version of a native P2P live streaming application.

3. DECADE Client API

In order to simplify the usage of DECADE server for P2P application clients, we provide a set of APIs. On top of these APIs, a P2P application client can develop its application-specific control and data distributed policies to utilize DECADE servers.

There are five basic APIs:

- o **Get_Object**: to get an object from a DECADE server with an authorized token. The get operation can be classified into two categories: Local Get and Remote Get. Local Get is to get an object from a local DECADE server. Remote Get is to use an application's local DECADE server to indirectly get an object from a remote DECADE server. The object will first passed to the local DECADE server, then return to the application client
- o **Put_Object**: to store an object into a DECADE server with an authorized token. An application client can either store an object in its local DECADE server or in other clients' DECADE servers, if it has a authorized token. Both operates are directly store, we don't provide indirectly store (i.e. remote put).
- o **Delete_Object**: to delete an object in a DECADE server explicitly with an authorized token. Know that an object can be deleted implicitly by setting a expired time or a specific TTL.
- o **Status_Query**: to query current status of an application itself, including a list of stored objects, resource usage, and so on. An application cannot query others' status.
- o **Generate-Token**: to generate an authorized token. The token can be used to access an application client's local DECADE server, or passed to other clients to allow others to access its DECADE server.

4. DECADE Integration of P2P LiveStreaming Client

We integrate DECADE client API with a P2P live streaming application, in order that clients of this P2P live streaming application can easily utilize DECADE server in their data transport processes.

4.1. DECADE Client Plug-in

DECADE Client Plug-in is a plug-in component, on top of DECADE client APIs, designed for application clients to use DECADE server. This plug-in component serves as an application-specific interface between a particular application and DECADE servers. In this test, we employed P2P live streaming application to leverage this plug-in.

4.2. DECADE Integration Architecture

The architecture of the P2P live streaming and DECADE integration is in Figure 1:

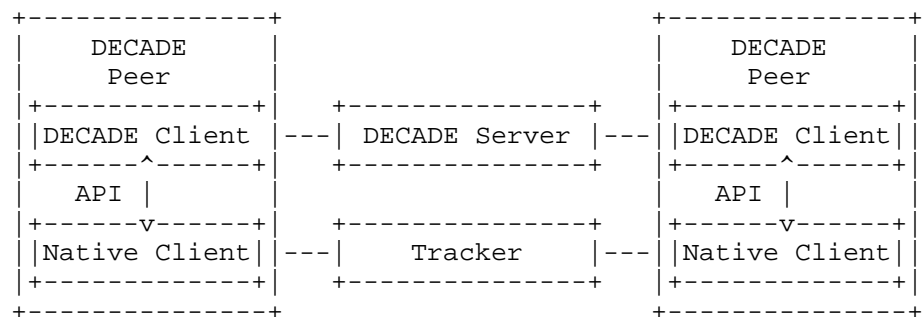


Figure 1

A DECADE integrated P2P live streaming client uses DECADE client plug-in to communicate with its DECADE server and access data between itself and DECADE servers. It uses an additional modified message protocol, as well as its original P2P message protocols to connect with other peers, exchange control messages.

4.2.1. Data Access

DECADE client plug-in is called whenever the application want to get data objects from (or put data objects into) DECADE servers. Every data object transferred between DECADE server and original P2P live streaming client must go through this plug-in. Neither the DECADE server and the original client knows each other. A Data object is a transfer unit of data, between DECADE servers and application clients. It the size of data object can be application-customized, according to variable requirements of performance or sensitive factors (e.g. low latency, high bandwidth utilization).

4.2.2. Message Control

Control and Data plane decoupling is a design principle of DECADE. Control messages are propagated in an original P2P way. Besides, original P2P control messages, a modified message protocol is needed for DECADE authorized token delivery. By exchanging DECADE authorized tokens, P2P live streaming clients can retrieve or store data objects into or from other clients' DECADE servers.

5. Test Environment and Settings

In order to demo our DECADE integrated P2P live streaming application, we conduct an test in Amazon EC2 and PlanetLab to show the performance of the system. For comparison, we conduct two tests, DECADE integrated P2P live streaming and native P2P live streaming, in the same environment use the same setting.

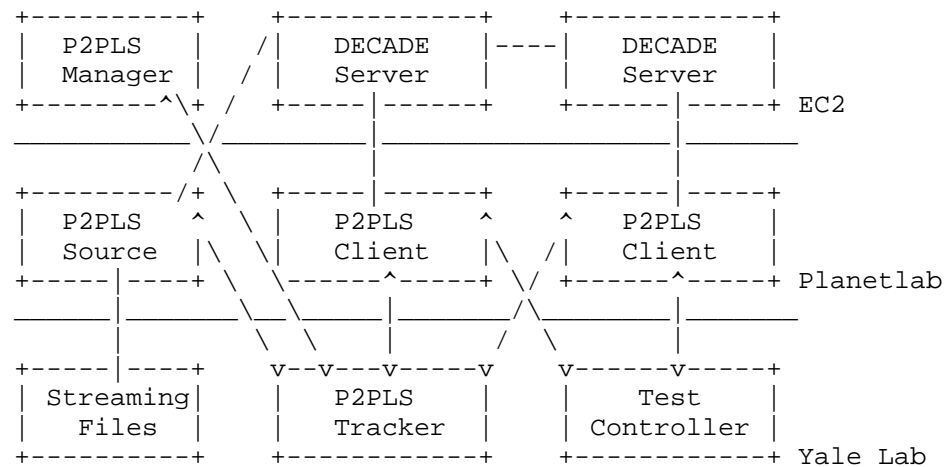
5.1. Test Settings

Our test ran on a wide-spread area and a diverse platforms, including a famous commercial cloud platform and a well-known testbed PlanetLab. The environment settings are as following:

- o EC2 Regions: we setup DECADE servers in Amazon EC2 cloud, including all four regions around the world, US east, US west, Europe and Asia.
- o PlanetLab: we run our P2P live streaming clients (both DECADE integrated and native clients) on PlanetLab of a wild-spread area.
- o Arrival pattern: we let all the clients join at the same time to simulate a flash crown scenario.
- o Total Bandwidth: for fair comparison, we set the system's total supply bandwidth to be exact the same in both test.

5.2. Platforms and Components

In our test, we have different functional components running in different platforms, including DECADE servers, P2P live streaming clients (DECADE integrated or Native), Tracker, Source server and Test Controller, as shown in Figure 2.



P2PLS represents to P2P LiveStreaming.

Figure 2

5.2.1. EC2 DECADE Server

DECADE Servers ran on Amazon EC2 small instances, with bandwidth constraint.

5.2.2. PlanetLab P2P LiveStreaming Client

Both DECADE integrated and Native P2P live streaming clients run in planetlab spreading in different locations in the world. The DECADE integrated P2P live streaming clients connect to a closest DECADE server according to its Geo-location distance of the servers. DECADE integrated P2P live streaming clients use their DECADE servers to upload to neighbors, instead of their own "last-mile" bandwidth.

5.2.3. Tracker

A native P2P live streaming tracker ran in our lab to serve for both DECADE integrated clients and native clients during the test.

5.2.4. Source Server

A native P2P live streaming source server ran in our lab to serve for both DECADE integrated clients and native clients during the test. The capacity of source is equivalently constrain for both cases.

5.2.5. Test Controller

Test Controller is a manager to control all machines behaviors in both EC2 and PlanetLab during the test.

6. Performance Analysis

During the test, DECADE integrated P2P live streaming clients achieve a better performance in the results.

6.1. Performance Metrics

- o Startup Delay: the time from a peer joins the channel to the moment it starts to play.
- o Piece Missed Rate: number of pieces a peer missed in the playing buffer to total number of pieces.
- o Freeze Times: number of a peer freeze during playing.
- o Average Peer Uploading Rate: Average uploading bandwidth of a peer per second.

6.2. Result and Analysis

- o Startup Delay: In the test, DECADE integrated P2P live streaming clients startup around 35~40 seconds. some of them startup at about 10 seconds. Native P2P live streaming clients startup around 110~120 seconds. less than 20% of them startup within 100 seconds.
- o Piece Missed Rate: In the test, both DECADE integrated P2P live streaming clients and native P2P live streaming clients achieved a good performance in pieces missed rate. Only about 0.02% of total pieces missed in both cases.
- o Freeze Times: In the test, native P2P live streaming clients suffered from 40% more freeze than DECADE Integrated P2P live streaming clients.
- o Average Peer Uploading Rate: In the test, according to our settings, DECADE integrated P2P live streaming clients had no upload in their "last-mile" access network. More than 70% of peers uploaded much more than streaming rate. That is to say, much uploading bandwidth are waste during data transport.

7. Security Considerations

This document does not contain any security considerations.

8. IANA Considerations

This document does not have any IANA considerations.

9. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Authors' Addresses

Lijiang Chen
Yale University

Email: lijiang.chen@yale.edu

Hongqiang Liu
Yale University

Email: hongqiang.liu@yale.edu

DECADE
Internet-Draft
Intended status: Informational
Expires: April 25, 2011

Z. Huang
X. Chen
HUAWEI Technologies
October 22, 2010

An integration example of DECADE system
draft-huang-decade-integration-example-01

Abstract

This document represents an integration experiment of DECADE. In the experiment, we use Vuze as the application which uses DECADE server for content distribution. This document describes the framework, message flow, environment settings and test steps. We also analyze the performance benefit of the experiment.

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 April 25, 2011.

Copyright Notice

Copyright (c) 2010 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. Vuze Client Design	4
3.1. DECADE-Enabled Vuze architecture Design	4
3.2. DECADE-Enabled Vuze Communication Procedure	5
4. Test Environment Setting	6
4.1. Test Platforms	7
4.1.1. DECADE Server	7
4.1.2. Vuze Client with DECADE Plugin	7
4.1.3. Remote Controller	8
4.1.4. Tracker	8
4.1.5. PL Manager	8
4.1.6. HTTP Server	8
4.2. Benchmark Setting	8
4.3. Test Steps	9
5. Performance Analysis	9
6. Security Considerations	11
7. IANA Considerations	11
8. Acknowledgments	11
9. References	11
9.1. Informative References	11
9.2. References	11
9.3. references	11
9.4. References	11
9.5. References	12
Authors' Addresses	12

1. Introduction

DECADE[draft-ietf-decade-problem-statement-00] is an architecture that provides applications with access and resource control to in-network storage.

This draft introduces an integration example of DECADE architecture. In our DECADE example system, the core component includes DECADE server and DECADE client. DECADE server running at Linux platform is designed to support data reading and writing for DECADE clients. For DECADE client, we choose an open source P2P client software named Vuze[Vuze] which supports the user defining plugin to extend software function. We designed a plugin named DECADE plugin to realize the DECADE function. Test Environment and performance of DECADE-Enable Vuze are described in the draft after introducing the system architecture and main message flow.

Please note that DECADE example system described in the draft is only an example of possible implementation. And Vuze presented in this draft is only one of many P2P applications. DECADE can support other applications, for example live streaming. We only show Vuze Integration experiment in the document.

2. Terminology

P2P: Peer-to-Peer computing or networking is a distributed application architecture that partitions tasks or work loads between peers. Peers are equally privileged, equipotent participants in the application.

Vuze: an open source P2P application, which uses BitTorrent protocol for message and data exchanging. Vuze provides a set of interfaces which support users to develop particular extensions.

DECADE Plugin: a plugin built into Vuze to realize DECADE functions including getting/putting data from/to DECADE server and redirection .

DECADE-Enabled Vuze: a Vuze client that is compatible with DECADE by implementing DECADE plugin and DECADE plugin is enabled.

Native Vuze: a Vuze client without DECADE plugin or with DECADE plugin but DECADE function is disabled.

DECADE server: a server software which runs at Linux system, providing the in-network storage function for DECADE Plugin to get and put the data. It's an implementation example of in-network

storage.

Remote Controller: an controller which can control every Vuze client to start or stop downloading tasks It also has the function of collecting statistics information of each Vuze client. It is a main operating platform.

3. Vuze Client Design

Note that Vuze client is classified into two different kinds - Native Vuze and DECADE-Enabled Vuze. When running Native Vuze, it behaves as usual BitTorrent client: some Vuze clients upload the data for other Vuze clients to download. When using DECADE-Enabled Vuze, the communication and data exchange process is different. The DECADE server will provide the data for Vuze clients to download. It means that uplink traffic can be reduced sharply and download performance can be improved. It is beneficial for ISPs to save the last-mile uplink bandwidth.

3.1. DECADE-Enabled Vuze architecture Design

DECADE plugin is one core part of our demo system. It has several interfaces with other components as following:

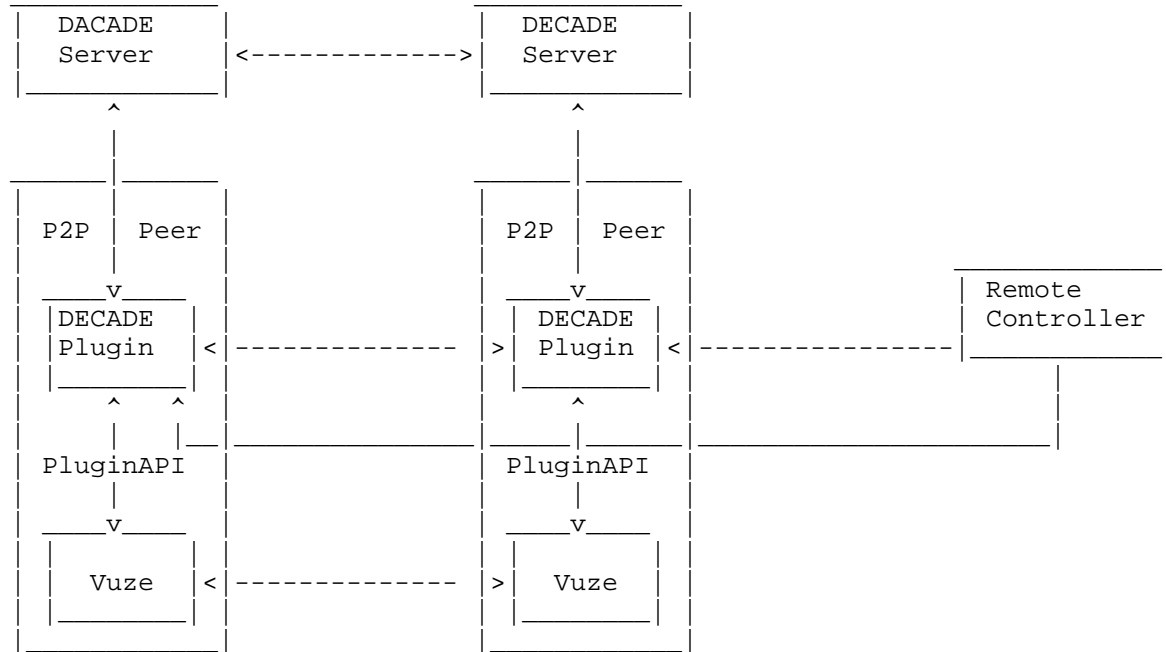
Interface between DECADE plugin and DECADE server: DECADE plugin can upload and download the data from DECADE server. It also includes other functions such as user registration, application registration etc.

Interface between DECADE plugin and Vuze client: DECADE plugin can register the listener to intercept the BitTorrent message such as "BT_Request" message from or to Vuze client, encapsulate the data from DECADE server into "BT_Piece" message and put the "BT_Piece" message into incoming message queue, read the block/piece data from the disk when seeding (to upload the data to DECADE server), and start or stop the download task, all these functions are supported in the Plugin API provided by Vuze.

Interface between DECADE plugins: When DECADE plugin intercepts the "BT_Request" message from other Vuze clients, local DECADE plugin sends "Redirect" message to remote DECADE plugin to authorize it to download the piece data from the DECADE server.

Interface between DECADE plugin and Remote Controller: DECADE plugin registers with Remote Controller after starting up, Remote Controller can control all the Vuze clients to start, stop or resume the download task through DECADE plugins.

The system architecture is as follow:



3.2. DECADE-Enabled Vuze Communication Procedure

A DECADE plugin can change the data path of BitTorrent download by using a "Redirect" message.

The detailed communication procedure is as following:

0 When each client starts the download task ,it will try to connect the tracker to get peer list and then send "BT_Request" message to other peers in peer list.

0 If the DECADE plugin is enabled, then it will intercept the incoming "BT_Request" message from other Vuze clients, and then reply with a "Redirect" message which includes DDECADE server's address, authorization token and so on to the requester.

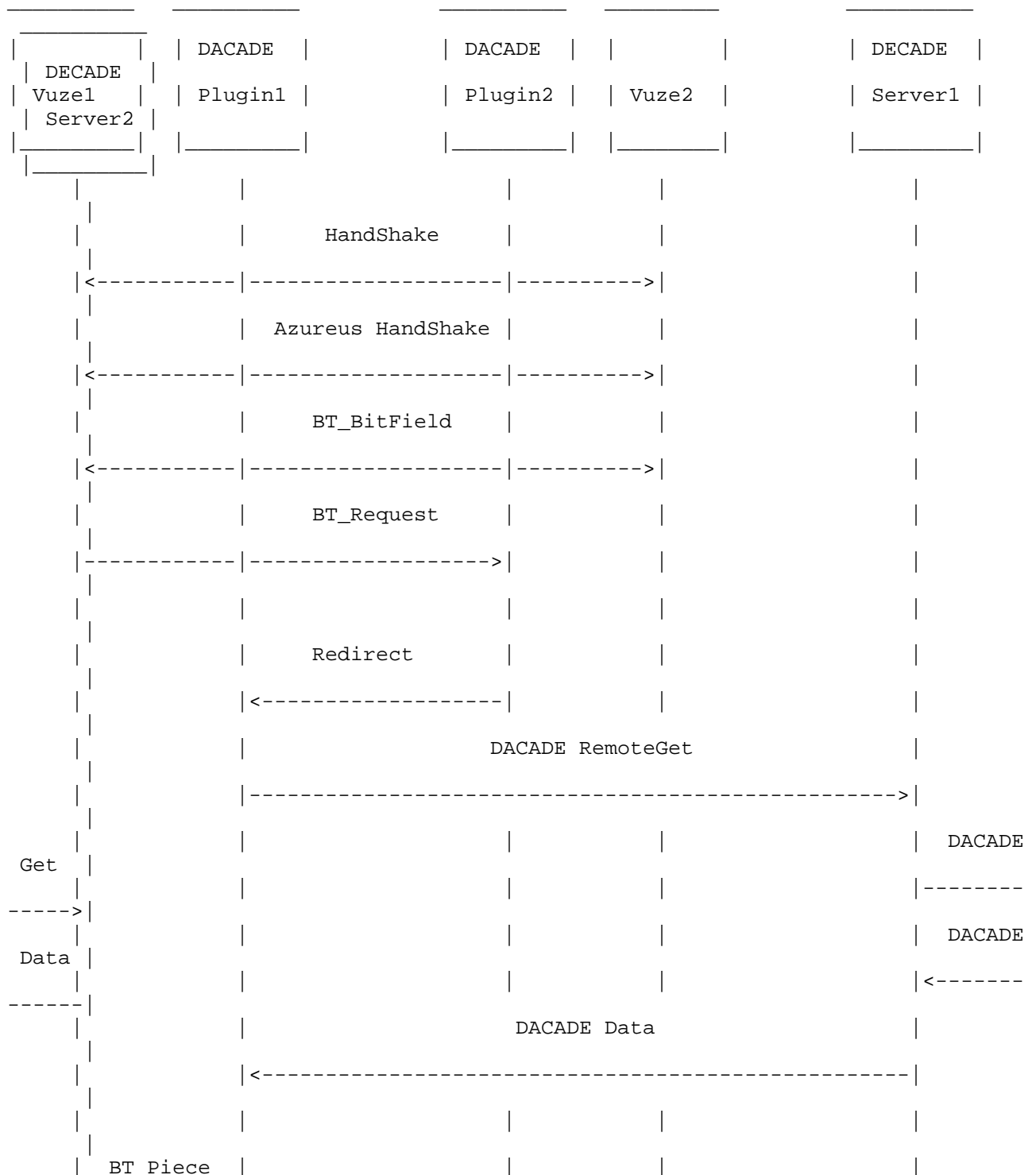
0 When a DECADE plugin receives a "Redirect" message, it will connect to the DECADE server according to message context and send "Remote Get" message to the DECADE server to request the data, then wait for response.

0 When a DECADE server receives a "Remote Get" message, it will check the server IP address in the message, Case 1: if the address equals

to its own IP address, then it will send the data to the requester from its local disk/memory; Case 2: if the address is not equal to its own IP address, then it will send the "Remote Get" message to that address, to fetch the data, and then send the data to the requester. The data will be cached in the server locally for use by other requesters.

0 When a DECADE plugin obtains the data, it will encapsulate the data into a standard "BT_Piece" message and send to Vuze client, then the client can write the data block into storage.

The detailed communication diagram is as follow:



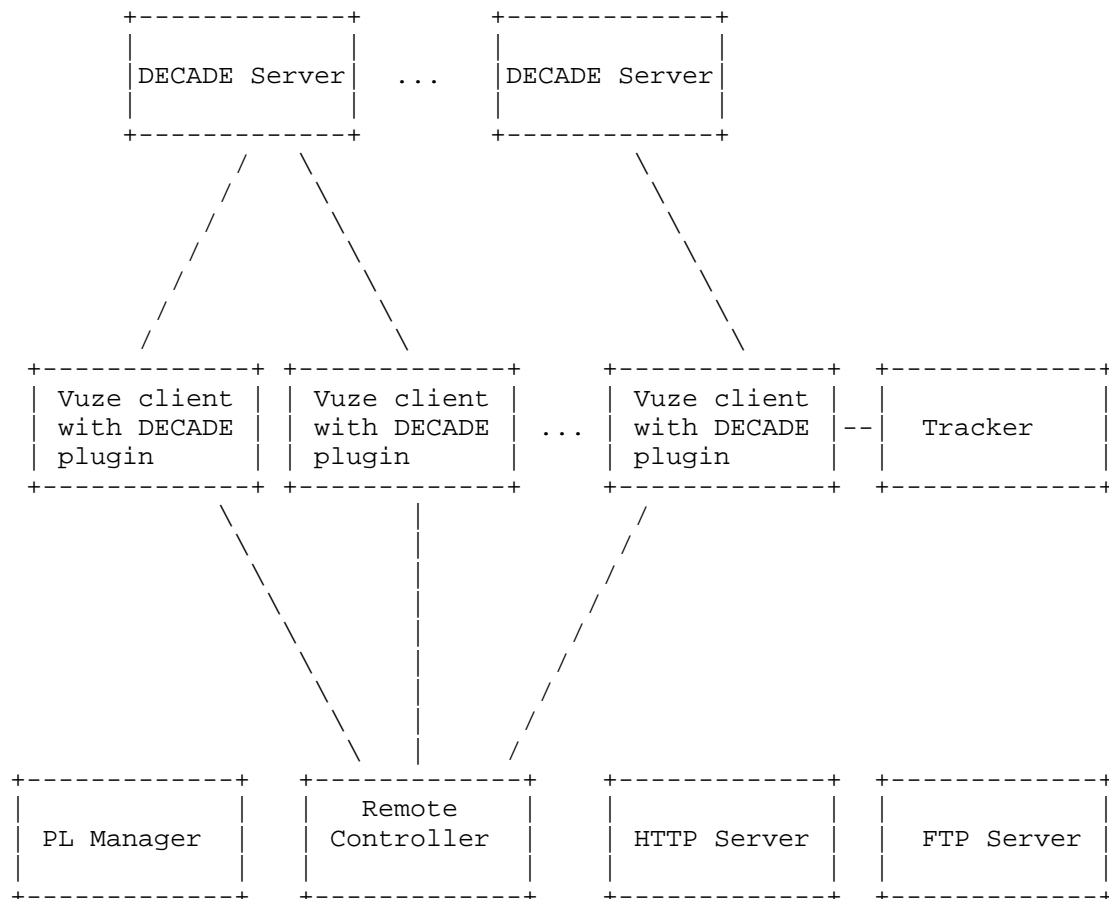


4. Test Environment Setting

Test environment includes Vuze client, tracker, DECADE Plugin, DECADE Servers and other supportive components such as HTTP Server, FTP Server and Remote Controller. For performance comparison of Native Vuze and DECADE-Enabled Vuze, we set the same amount of total network

bandwidth resource in these two test cases. It means the total uplink bandwidth of all the Vuze clients in the native Vuze test case is equal to the bandwidth of DECADE server in DECADE-Enabled Vuze test case.

4.1. Test Platforms



4.1.1. DECADE Server

DECADE Server runs at Amazon EC2 [Amazon-EC2]small instances.

4.1.2. Vuze Client with DECADE Plugin

Vuze clients include seeding client and leechers. Leechers run at PlanetLab.[PlanetLab] and the seeding client runs at Window 2003 server. DECADE Plugin will be automatically loaded and run after

Vuze client starts up.

4.1.3. Remote Controller

Remote controller can list all the Vuze clients in user interface and control them to download the specific BitTorrent file. It runs at Window 2003 server.

4.1.4. Tracker

Vuze client provides tracker capability, so we did not deploy our own tracker. Vuze embedded tracker is enabled when making a torrent file, the seeding client is also the tracker.

4.1.5. PL Manager

PL Manager [PlanetLab-experiment-manager] is a tool developed by University of Washington, which presents a simple GUI to control PlanetLab nodes and perform common tasks such as:

- O Selecting nodes for your slice.
- O Choosing nodes for your experiment based on CoMon information about the nodes.
- O Reliably deploying your experiment files.
- O Executing commands / sets of commands on every node in parallel.
- O Monitoring the progress of the experiment as a whole, as well as viewing console output from the nodes.

4.1.6. HTTP Server

Torrent file will be put in the HTTP Server and the leechers will retrieve the torrent file from HTTP Server after receiving the download command from Remote Controller. We use Apache Tomcat which is an open source software as HTTP Server.

4.2. Benchmark Setting

For the performance comparison of Native Vuze and DECADE-Enabled Vuze, the same system bandwidth is provided through some parameter settings. In the Native Vuze, the system bandwidth is defined as the amount of the uplink bandwidth from all the Vuze clients. The maximum upload bandwidth (B_u) is configured to every Vuze client before the test. Suppose the number of the Vuze clients is N , the system bandwidth is $B_u * N$. In the DECADE-Enabled Vuze case, the same

bandwidth ($B_u * N$) is configured to the corresponding Ethernet port of DECADE Server.

4.3. Test Steps

Test steps of DECADE-Enabled Vuze is as follows:

- O Start DECADE Server, Remote Controller and all the Vuze clients.
- O One of DECADE Vuze clients seeds, generates the torrent file. DECADE Plugin automatically uploads the data to DECADE Server.
- O Manually put the torrent file into home directory of HTTP server. If the default directory of torrent file in Vuze client is set to the home directory of HTTP server, this step can be ignored.
- O Remote Controller sends the download command to other DECADE-Enabled Vuze clients (leechers).
- O Leechers fetch torrent file from HTTP server.
- O Leechers request and get peer list from the tracker.
- O Leechers send BT_Request message to peers and get Redirect messages from peers.
- O Leechers get objects from DECADE Servers and finish the download.
- O Collecting and analyzing the log from all the Vuze clients.

Test steps of Native Vuze is similar to DECADE-Enabled Vuze. The difference is that Native Vuze client does not use DECADE server to put and get data.

5. Performance Analysis

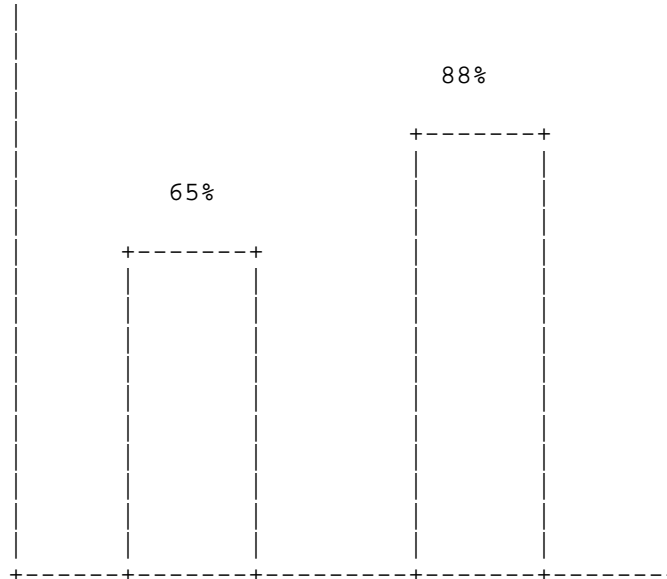
Performance advantage is shown according to the test result of experiment:

- O There is no upload data traffic from Vuze clients in the DECADE-Enabled Vuze experiment, but in the Native Vuze experiment, the upload data traffic from Vuze clients is same as download data traffic. Bandwidth resource is reduced in the last mile in the DECADE-Enabled Vuze experiment. The test result is illustrated in the following table. The download traffic and upload traffic include

data traffic and signal traffic. For the upload traffic in the DECADE-enabled Vuze, the data traffic is zero so the all traffic is signal overhead. Though we used the same torrent file in the two cases, but the number of Vuze client was not the same because the nodes in the Planet-lab are not always stable, so the download traffic is much different in two cases.

	Download traffic	Upload traffic
DECADE-Enabled Vuze	480MB	12MB
Native Vuze	430MB	430MB

O Higher system resource efficiency in the DECADE-Enabled Vuze experiment, system resource efficiency is defined as the ratio of system download rate to the total system bandwidth. The test result is illustrated in the following figure.



Native Vuze DECADE-Enabled Vuze

6. Security Considerations

This draft does not introduce any security considerations.

7. IANA Considerations

This document does not have any IANA Considerations.

8. Acknowledgments

The authors would like to thank Haibin Song, Richard Yang and Ning Zong for comments and contributions to this document, and also thanks Paul Gardner, Yudong Kang, Tao Jiang, Jinglong Liu and Cixiang Lei for programming support.

9. References

9.1. Informative References

[draft-ietf-decade-problem-statement-00]
Song, H., Zong, N., Yang, Y., and R. Alimi, "DECoupled Application Data Enroute (DECADE) Problem Statement", August 2010.

9.2. References

[PlanetLab]
"PlanetLab", <http://www.Planet-lab.org>.

9.3. references

[PlanetLab-experiment-manager]
Computer Science and Engineering, University of Washington, "PlanetLab experiment manager", <http://www.cs.washington.edu/research/networking/cplane/>.

9.4. References

[Amazon-EC2]
Amazon, "Amazon Elastic Compute Cloud", <http://aws.amazon.com/ec2/>.

9.5. References

[Vuze] "Vuze", <http://www.vuze.com>.

Authors' Addresses

Z Huang
HUAWEI Technologies

Email: hpanda@huawei.com

X Chen
HUAWEI Technologies

Email: chen.xiaohui@huawei.com

DECADE
Internet-Draft
Intended status: Standards Track
Expires: February 25, 2011

H. Song
N. Zong
Huawei
Y. Yang
R. Alimi
Yale University
August 24, 2010

DECoupled Application Data Enroute (DECADE) Problem Statement
draft-ietf-decade-problem-statement-00

Abstract

Peer-to-peer (P2P) applications have become widely used on the Internet today and make up a large portion of the traffic in many networks. In P2P applications, one technique for reducing the total amount of P2P traffic is to introduce storage capabilities within the network. Traditional caches (e.g., P2P and Web caches) provide such storage, but they are complex (due to explicitly supporting individual P2P application protocols) and they do not allow users to manage access to content in the cache. For example, Content Providers cannot easily control access and resource usage policies to satisfy their own requirements. This document discusses the introduction of in-network storage for P2P applications, shows the need for a standard protocol for accessing this storage, and identifies the scope of this protocol. The accessing protocol can also be used by other applications with similar requirements.

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 February 25, 2011.

Copyright Notice

Copyright (c) 2010 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 and Concepts	4
3. The Problems	4
3.1. P2P infrastructural stress and inefficiency	5
3.2. P2P cache: a complex in-network storage	5
3.3. Ineffective integration of P2P applications and in-network storage	6
4. DECADE as an In-network Storage Capability	7
4.1. Data access	9
4.2. Authorization	10
4.3. Resource control	10
5. Usage Scenarios	10
5.1. BitTorrent	10
5.2. Content Publisher	11
5.3. Data Transfer Scenarios	12
5.3.1. Both Sender and Receiver Accounts are Used	12
5.3.2. Only Sender's Storage Account is Used	13
5.3.3. Only Receiver's Storage Account is Used	13
5.3.4. No Storage Accounts are Used	13
6. Security Considerations	14
6.1. Denial of Service attack	14
6.2. Copyright and Legal issues	14
7. IANA Considerations	14
8. Acknowledgments	14
9. References	15
9.1. Normative References	15
9.2. Informative References	15
Authors' Addresses	16

1. Introduction

P2P applications, including both P2P streaming and P2P filesharing applications, make up a large fraction of the traffic in many Internet networks today. One way to reduce bandwidth usage by P2P applications is to introduce storage capabilities in the networks. Allowing P2P applications to store and retrieve data from inside networks can reduce traffic on the last-mile uplink, as well as backbone and transit links [I-D.weaver-alto-edge-caches].

P2P caches provide in-network storage and have been deployed in some networks. But the current P2P caching architecture poses challenges to both P2P cache vendors and P2P application developers. For P2P cache vendors, it is challenging to support a number of continuously evolving P2P application protocols, due to lack of documentation, ongoing protocol changes, and rapid introduction of new features by P2P applications. For P2P applications, closed P2P caching systems limit P2P applications to effectively utilize in-network storage. In particular, P2P caches typically do not allow users to explicitly store content into in-network storage. Neither do they allow users to implement control over the content that have been placed into the in-network storage.

Both of these challenges can be effectively addressed by using an open, standard protocol to access in-network storage. P2P applications can store and retrieve content in the in-network storage, as well as control resources (e.g., bandwidth, connections) consumed by peers in a P2P application. As a simple example, a peer of a P2P application may upload to other peers through its in-network storage, saving its usage of last-mile uplink bandwidth.

In this document, we distinguish between two functional components of the native P2P application protocol: signaling and data access. Signaling includes operations such as handshaking and discovering peer and content availability. The data access component transfers content from one peer to another.

With DECADE, P2P applications can still use their native protocols for signaling and data transport. However, they may use a standard protocol for data access incorporating in-network storage, and fall back to their native data transport protocols if in-network storage is not available or not used.

In essence, an open, standard protocol to access in-network storage provides an alternative mechanism for P2P application data access that is decoupled from P2P application control and signaling. This decoupling leads to many advantages, which is explained further in Section 4.

2. Terminology and Concepts

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].

The following terms have special meaning in the definition of the in-network storage system.

In-network Storage: A service inside a network that provides storage and bandwidth to network applications. In-network storage may reduce upload/transit/backbone traffic and improve network application performance.

IAP (In-network storage Access Protocol): a standard protocol that is spoken between P2P applications and in-network storage. The protocol may also be used between entities implementing the in-network storage service. IAP may be a new protocol or existing protocol with extensions.

P2P Cache (Peer to Peer Cache): a kind of in-network storage that understands the signaling and transport of specific P2P application protocols, it caches the content for those specific p2p applications in order to serve peers and reduce traffic on certain links.

Content Publisher: An entity that originates content to consumers.

3. The Problems

The emergence of peer-to-peer (P2P) as a major type of network applications (esp. P2P file sharing and streaming apps) has led to substantial opportunities. The P2P paradigm can be utilized in designing highly scalable and robust applications at low cost, compared with traditional client-server paradigms. For example, CNN [CNN] reported that P2P streaming by Octoshape played a major role in its distribution of the historical inauguration address of President Obama. PPLive, one of the largest P2P streaming vendors, is able to distribute large-scale, live streaming programs to more than 2 million users with only a handful of servers.

However, P2P applications also face substantial design challenges. A particular problem facing P2P applications is the substantial stress that they place on the network infrastructure. Also, lacking of infrastructure support can lead to unstable P2P application performance during peer churns and flash crowd. Below we elaborate on the problems in more detail.

3.1. P2P infrastructural stress and inefficiency

A particular problem of the P2P paradigm is the stress that P2P application traffic places on the infrastructure of Internet service providers (ISP). Multiple measurements (e.g., [ipoque]) have shown that P2P traffic has become a major type of traffic on some networks. Furthermore, network-agnostic peering leads to unnecessary traversal across network domains or spanning the backbone of a network, leading to network inefficiency [I-D.ietf-alto-problem-statement].

Recently, the IETF Application Layer Traffic Optimization (ALTO) Working Group was formed to provide P2P applications with network information so that they can perform better-than-random initial peer selection [I-D.ietf-alto-problem-statement]. However, there are limitations on what ALTO can achieve alone. For example, network information alone cannot reduce P2P traffic in access networks, as the total access upload traffic is equal to the total access download traffic in a pure P2P system. On the other hand, it is reported that P2P traffic is becoming the dominating traffic on the access networks of some networks, reaching as high as 50-60% at the down-links and 60-90% at the uplinks ([DCIA], [ICNP], [ipoque.P2P_survey.], [P2P_file_sharing]). Consequently, it becomes increasingly important to complement the ALTO effort and reduce upload access traffic, in addition to cross-domain and backbone traffic.

The IETF Low Extra Delay Background Transport (LEDBAT) Working Group is focusing on techniques that allow large amounts of data to be consistently transmitted without substantially affecting the delays experienced by other users and applications. It is expected that some P2P applications would start using such techniques, thereby somewhat alleviating the perceivable impact (at least on other applications) of their high volume traffic. However, such techniques may not be adopted by all P2P applications. Also, when adopted, these techniques do not remove all inefficiencies, such as those associated with traffic being sent upstream as many times as there are remote peers interested in getting the corresponding information. For example, the P2P application transfer completion times remain affected by potential (relatively) slow upstream transmission. Similarly, the performance of real-time P2P applications may be affected by potential (relatively) higher upstream latencies.

3.2. P2P cache: a complex in-network storage

An effective technique to reduce P2P infrastructural stress and inefficiency is to introduce in-network storage. For example, in [I-D.weaver-alto-edge-caches], the author demonstrates clearly the potential benefits of introducing in-network storage to improve

network efficiency and thus reduce network infrastructure stress.

In the current Internet, in-network storage is introduced as P2P caches, either transparently or explicitly as a P2P peer. To provide service to a specific P2P application, the P2P cache server must support the specific signaling and transport protocols of the specific P2P application. This can lead to substantial complexity for the P2P Cache vendor.

First, there are many P2P applications on the Internet (e.g., BitTorrent, eMule, Flashget, and Thunder for file sharing; Abacast, Kontiki, Octoshape, PPLive, PPStream, and UUSee for P2P streaming). Consequently, a P2P cache vendor faces the challenge of supporting a large number of P2P application protocols, leading to product complexity and increased development cost.

Furthermore, a specific P2P application protocol may be evolving continuously, to add new features or fix bugs. This forces a P2P cache vendor to continuously update to track the changes of the P2P application, leading to product complexity, high cost, and low reliability.

Third, many P2P applications use proprietary protocols or support end-to-end encryption. This can render P2P caches ineffective.

3.3. Ineffective integration of P2P applications and in-network storage

As P2P applications evolve, it becomes increasingly clear that they will need in-network resources to provide positive user experiences. For example, multiple P2P streaming systems seek additional in-network resources during flash crowd, such as just before a major live streaming event. In asymmetric networks when the aggregated upload bandwidth of a channel cannot meet the download demand, a P2P application may seek additional in-network resources to maintain a stable system.

A requirement by some P2P applications in using in-network infrastructural resources, however, is flexibility in implementing resource allocation policies. A major competitive advantage of many successful P2P systems is their substantial expertise in how to most efficiently utilize peer and infrastructural resources. For example, many live P2P systems have their specific algorithms in selecting the peers that behave as the more stable, higher bandwidth sources. They continue to fine-tune such algorithms. In other words, in-network storage should export basic mechanisms and allow as much flexibility as possible to P2P applications to implement specific policies. This conforms to the end-to-end systems principle and allows innovation and satisfaction of specific business goals. Existing techniques for

P2P application in-network storage lack these capabilities.

4. DECADE as an In-network Storage Capability

The objective of this working group is to design DECADE, an in-network storage protocol to address the problems discussed in the preceding section.

DECADE will provide access to storage and data transport services in the network to P2P applications to improve their efficiency and reduce the stress on the network infrastructure. Unlike the existing P2P caching architecture, DECADE is a standard interface for various P2P applications (both content publishers and end users) to access in-network storage. This decoupling of P2P data access from P2P application control and signaling reduces the complexity of in-network storage services. Furthermore, DECADE provides basic access mechanisms and allows P2P applications to implement flexible policies to create an ecosystem for application innovation and various business goals. Besides that, it also improves the availability of P2P contents because the in-network storage is always-on.

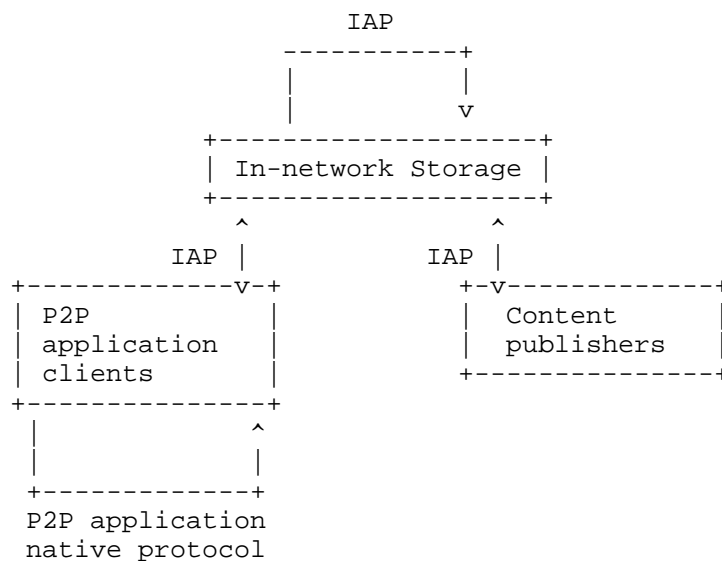


Figure 1 Overview of protocol interaction between DECADE elements

Specifically, the main component of DECADE is the in-network storage access protocol (IAP), which is a standard, P2P-application-agnostic protocol for different P2P applications to access in-network storage. IAP defines a set of commands that P2P application elements can issue

to in-network storage to store and retrieve data. IAP includes the following functionalities:

- (1) Data access provides read/write by users (e.g., P2P application clients and content publishers) to the corresponding in-network storage and between entities implementing the in-network storage;
- (2) Authorization implements access control to resources provided by in-network storage;
- (3) Resource control allows users to implement application policies for the corresponding in-network storage.

Note that DECADE is independent of current IETF work on P2P, e.g. P2PSIP, ALTO, PPSP. For example, peers discovered by either RELOAD or ALTO can all use DECADE to share data.

The Peer to Peer Streaming Protocol effort in the IETF is investigating specification of signaling protocols (called PPSP protocols) for multiple types of entities (e.g. intelligent endpoints, caches, content distribution network nodes, and/or other edge devices) to participate in P2P streaming systems in both fixed and mobile Internet. As discussed in the PPSP problem statement document [I-D.zhang-ppsp-problem-statement], one important PPSP use case is the support of an in-network edge Cache for P2P Streaming. However, this approach to providing in-network cache has different applicability, different objectives and different implications for the in-network cache operator. A DECADE service can be used for any application transparently to the DECADE in-network storage operator: it can be used for any P2P Streaming application (whether it supports PPSP protocols or not), for any other P2P application, and for non P2P applications that simply want to benefit from in-network storage. So with DECADE the operator is providing a generic in-network storage service that can be used by any application without application involvement or awareness by the operator; in the PPSP cache use case, the cache operator is participating in the specific P2P streaming service.

DECADE and PPSP can both contribute independently, and (where appropriate) simultaneously, to making content available closer to peers. Here are a number of example scenarios:

A given network supports DECADE in-network storage, and its CDN nodes do not participate as PPSP Peers for a given "stream" (e.g. say because no CDN arrangement has been put in place between the Content Provider and the considered network provider). In that case, PPSP Peers will all be "off-net" but will be able to use DECADE in-network storage to exchange chunks.

A given network does not support DECADE in-network storage, and (some of) its CDN nodes participate as PPSP Peers for a given "stream" (e.g. say because an arrangement has been put in place between the Content Provider and the considered network provider). In that case, the CDN nodes will participate as in-network PPSP Peers. The off-net PPSP Peers (ie end users) will be able to get chunks from the in-network CDN nodes (using PPSP protocols with the CDN nodes).

A given network supports DECADE in-network storage, and (some of) its CDN nodes participate as PPSP Peers for a given "stream" (e.g. say because an arrangement has been put in place between the Content Provider and the considered network provider). In that case, the CDN nodes will participate as in-network PPSP Peers. The off-net PPSP Peers (ie end users) will be able to get chunks from the in-network CDN nodes (using PPSP protocols with the CDN nodes) as well as be able to get chunks /share chunks using DECADE in-network storage populated (using IAP protocol) by PPSP Peers (both off-net end-users and in-network CDN Nodes).

While DECADE will focus on P2P applications, the solution is expected to be applicable in other contexts with similar requirements.

4.1. Data access

P2P application clients use the protocol to read data from an in-network storage, store data to an in-network storage, or remove data from an in-network storage. The data could be of various types. Existing protocols will be used wherever possible and appropriate to support DECADE's requirements. In particular, data storage, retrieval, and management may be provided by an existing IETF protocols. The WG will not limit itself to a single data transport protocol since different protocols may have varying implementation costs and performance tradeoffs. However, to keep interoperability manageable, a small number of specific, targeted, data transport protocols will be identified and used. If a protocol is found to be suitable but does not fully meet the requirements, then the protocol may need to be extended. The following considerations should be taken into account. But it might be a trade-off when making decision.

- o The protocol(s) should support deployments with a very large number of users without substantial increase to operational complexity for the storage provider.
- o The protocol(s) should be easy for applications to integrate with when they want to use it for P2P applications (e.g. file-sharing or streaming) or other content distribution applications.

4.2. Authorization

DECADE ensures that access to the in-network storage is subject to authorization by the user owning the in-network storage service. The authorization can take into account the user trying to access, the content, the time period, etc.

4.3. Resource control

A user uses the protocol to manage the resources on in-network storage that can be used by other peers, e.g., the bandwidth or connections. The resource control policies could be based on individual remote peers or a whole application.

5. Usage Scenarios

Usage scenarios are described from two perspectives. First, we introduce high-level use cases showing how P2P applications may utilize in-network storage. Second, we show how in more detail how users exchange data using IAP.

5.1. BitTorrent

BitTorrent may be integrated with DECADE to be more network efficient and reduce the bandwidth consumed on ISP networks. When a BitTorrent client uploads a block to peers, the block traverses the last-mile uplink once for each peer. With DECADE, however, the BitTorrent client may upload the block to its in-network storage. Peers may retrieve the block from the in-network storage, reducing the amount of data on the last-mile uplink.

We now describe in more detail how BitTorrent can utilize DECADE. For illustration, we assume that both the BitTorrent client (A) and its peer (B) utilize in-network storage. When A requests a block, it indicates that it would like the block stored in its in-network storage and provides the necessary access control. Instead of sending the 'piece' message with the desired block, peer B replies with a 'redirect' message indicating that the content should be retrieved from its own in-network storage and provides the necessary access control. If the peer B had not previously stored the content in its in-network storage, it uploads the block. A downloads the block into its own in-network storage from B's in-network storage, and finally A itself retrieves the block.

Note that this requires extensions to the BitTorrent protocol. While there are multiple ways to do so, this example assumes the native BitTorrent 'request' message is extended to carry additional

information and that a new 'redirect' message is added. Upload and download to/from in-network storage uses the standard IAP protocol.

This example has illustrated how utilizing DECADE can increase BitTorrent's network efficiency. First, notice that peer B does not utilize any uplink bandwidth if the block was already present in its in-network storage. Second, notice that the block is downloaded directly into A's in-network storage. When A wishes to share the block with another peer (say, peer C) that supports DECADE, it may upload directly from its in-network storage, again avoiding usage of the last-mile uplink.

This technique can be applied to other P2P applications as well. Since P2P applications use a standard for communicating with in-network storage, they no longer require in-network storage to explicitly support their protocol. P2P applications retain the ability to explicitly manage which content is placed in in-network storage, as well as access and resource control policies.

5.2. Content Publisher

Content Publishers may also utilize in-network storage. For example, consider a P2P live streaming application. A Content Publisher typically maintains a small number of sources, each of which distributes blocks in the current play buffer to a set of the P2P peers.

Consider a case where the Content Publisher owns an in-network storage account within ISP A. If there are multiple P2P peers within ISP A, the Content Publisher may utilize DECADE to distribute content to the peers.

First, the Content Publisher stores a block in the in-network storage, and then sends to each peer in ISP A the block's identifier and necessary access control. Second, each peer may then download from the Content Publisher's in-network storage.

In this example, the block is distributed in a more network efficient way (the content only traverses the ISP's interdomain link once), while the Content Publisher retains explicit control over access to the content placed in its own storage. The Content Publisher can remove content from its in-network storage when it is stale or needs to be replaced, and grant access and resources to only the desired peers.

Note that Content Publishers and individual peers can each use in-network storage. For example, after downloading content from the Content Publisher's in-network storage, peers may each utilize their

own in-networks storage similar to the usage scenario in Section 5.1. This can have the benefit of increased network efficiency, while Content Publishers and peers still retain control over content placed in their own in-network storage.

5.3. Data Transfer Scenarios

The previous usage scenarios have utilized the ability for peers to transfer data by storing and retrieving from in-network storage. This section describes in further detail an example solution of how DECADE can provide this capability.

In this section, we consider the case of a user B (the receiver) requesting data from user A (the sender). We use *Sa* to denote User A's storage account, and *Sb* to denote User B's storage account. Each user independently decides if its in-network storage account is used, so there are four cases.

When a user indicates that it wishes to use its in-network storage, it provides an access control token to the other user. The token is sent using the application's protocol. To simplify the illustration, we omit details of the access control from the detailed scenarios below.

5.3.1. Both Sender and Receiver Accounts are Used

This scenario is illustrated in Figure 2. B first requests an object from A using the application protocol indicating it wishes the object to be stored in *Sb*. A responds using the application protocol indicating that B should download the object from *Sa*. B sends a IAP request to *Sb* indicating that the object should be downloaded from *Sa*. *Sb* uses IAP to download from *Sa*, and finally, B downloads the object from *Sb* (also using IAP).

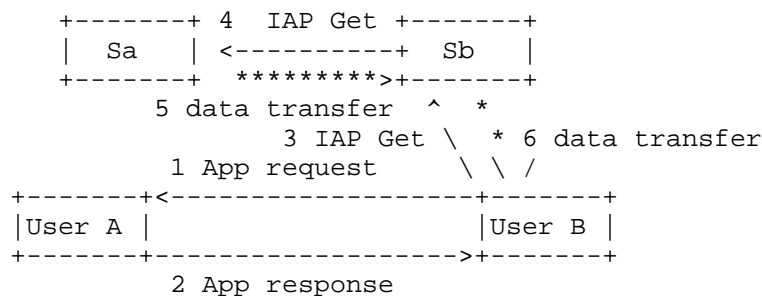


Figure 2: Usage Scenario 1 (Sender and receiver Accounts used)

5.3.2. Only Sender's Storage Account is Used

This scenario is illustrated in Figure 3. B requests an object from A using the application protocol. A responds using the application protocol indicating that B should download the object from Sa. Finally, B sends a IAP request to Sa to download the object.

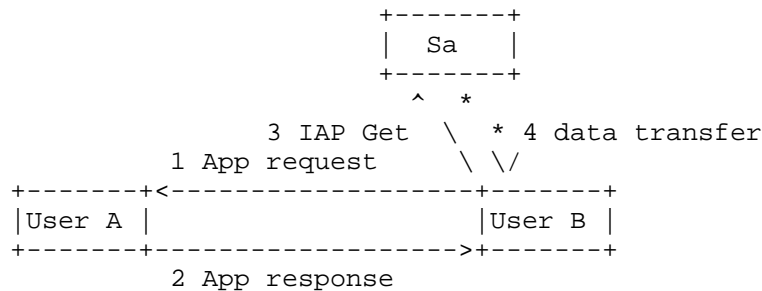


Figure 3: Usage Scenario 2 (Sender account used)

5.3.3. Only Receiver's Storage Account is Used

This scenario is illustrated in Figure 4. B requests an object from A using the application protocol indicating that it wishes the object to be stored in Sb. A stores the object in Sb (using IAP), and responds to B (using the application protocol) that it should download from Sb. B uses IAP to download the object from Sb.

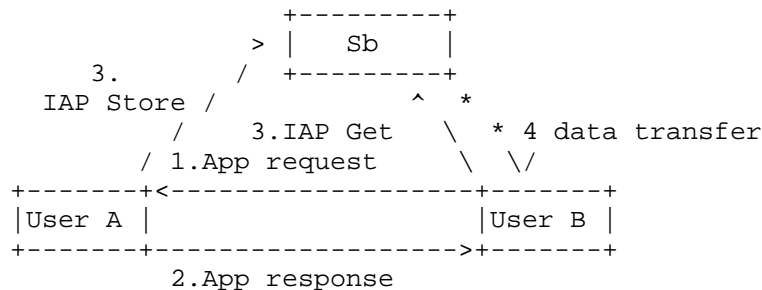


Figure 4: Usage Scenario 3 (Receiver account used)

5.3.4. No Storage Accounts are Used

This scenario is illustrated in Figure 5. In this scenario, the application protocol is used directly to send data. This scenario applies with one of the peers does not support IAP, or neither of the peers are using in-network storage.

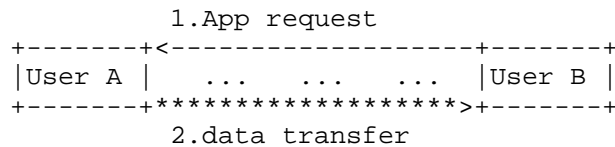


Figure 5: Usage Scenario 4 (No storage accounts used)

6. Security Considerations

There are multiple security considerations. We focus on two in this section.

6.1. Denial of Service attack

Without access control or resource control, an attacker can try to consume a large portion of the in-network storage, or exhaust the connections of the in-network storage to commit a Denial of Service (DoS) attack. Thus, access control and resource control mechanisms are mandatory. A resource control mechanism should be used to allow a user to allocate the resource in its in-network storage account to be utilized by other clients.

6.2. Copyright and Legal issues

Copyright and other laws may prevent the distribution of certain content in various localities. While in-network storage operators may adopt system-wide ingress or egress filters to implement necessary policies for storing or retrieving content, the specification and implementation of such policies (e.g., filtering and DRM) is outside of the scope of this working group.

7. IANA Considerations

There is no IANA consideration with this document.

8. Acknowledgments

We would like to thank the following people for contributing to this document:

David Bryan

Francois Le Faucheur

Hongqiang Law.

Kar Ann Chew

Richard Woundy

Roni Even

Yunfei Zhang

Yu-shun Wang

Yingjie Gu

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

[ipoque.com]

"http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009".

[I-D.ietf-alto-problem-statement]

Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", draft-ietf-alto-problem-statement-04 (work in progress), September 2009.

[I-D.weaver-alto-edge-caches]

Weaver, N., "Peer to Peer Localization Services and Edge Caches", draft-weaver-alto-edge-caches-00 (work in progress), March 2009.

[I-D.zhang-ppsp-problem-statement]

Zhang, Y., Zong, N., Camarillo, G., Seng, J., and Y. Yang, "Problem Statement of P2P Streaming Protocol (PPSP)", draft-zhang-ppsp-problem-statement-06 (work in progress), July 2010.

[RFC3720]

Satran, J., Meth, K., Sapuntzakis, C., Chadalapaka, M., and E. Zeidner, "Internet Small Computer Systems Interface (iSCSI)", RFC 3720, April 2004.

- [RFC5661] Shepler, S., Eisler, M., and D. Noveck, "Network File System (NFS) Version 4 Minor Version 1 Protocol", RFC 5661, January 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.
- [DCIA] <http://www.dcia.info>, "Distributed Computing Industry Association".
- [ipoque.P2P_survey.]
"Emerging Technologies Conference at MIT", Sept. 2007.
- [P2P_file_sharing]
Parker, A., "The true picture of peer-to-peer filesharing", July 2004.
- [ICNP] Wu, H., "Challenges and opportunities of Internet developments in China, ICNP 2007 Keynote", Oct. 2007.

Authors' Addresses

Haibin Song
Huawei
Baixia Road No. 91
Nanjing, Jiangsu Province 210001
P.R.China

Phone: +86-25-56622975
Email: melodysong@huawei.com

Ning Zong
Huawei
Baixia Road No. 91
Nanjing, Jiangsu Province 210001
P.R.China

Phone: +86-25-56622975
Email: zongning@huawei.com

Y. Richard Yang
Yale University

Email: yry@cs.yale.edu

Richard Alimi
Yale University

Email: richard.alimi@yale.edu

DECADE
Internet-Draft
Intended status: Informational
Expires: April 21, 2011

Y. Gu
Huawei
D. Bryan
Cogent Force, LLC / Huawei
Y. Yang
Yale University
R. Alimi
Google
October 18, 2010

DECADE Requirements
draft-ietf-decade-reqs-00

Abstract

The target of DECoupled Application Data Enroute (DECADE) is to provide an open and standard in-network storage system for applications, primarily P2P applications, to store, retrieve and manage their data. This draft enumerates and explains requirements, not only for store and retrieve, but also for data management, access control and resource control, that should be considered during the design and implementation of a DECADE system. These are requirements on the entire system; some of the requirements may eventually be implemented by an existing protocol with/without some extensions (e.g., the data transport level). A user of DECADE as a complete architecture would be guaranteed complete functionality.

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 [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 April 21, 2011.

Copyright Notice

Copyright (c) 2010 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 BSD License.

Table of Contents

1. Introduction	5
2. Terminology and Concepts	5
3. Requirements Structure	6
4. Protocol Requirements	7
4.1. Requirements	7
4.1.1. Overall Protocol Requirements	7
4.1.1.1. Application-independent API	7
4.1.1.2. Cross-platform Access	7
4.1.1.3. Connectivity Concerns	7
4.1.1.3.1. NATs and Firewalls	7
4.1.1.3.2. Connections to Clients	8
4.1.1.4. Error and Failure Conditions	8
4.1.1.4.1. Overload Condition	8
4.1.1.4.2. Insufficient Resources	8
4.1.1.4.3. Unavailable and Deleted Data	9
4.1.2. Transfer and Latency Requirements	9
4.1.2.1. Low-Latency Access	9
4.1.2.2. Indirect Transfer	9
4.1.2.3. Data Object Size	10
4.1.2.4. Communication among In-network Storage Elements	10
4.1.3. Data Access Requirements	10
4.1.3.1. Reading/Writing Own Storage	10
4.1.3.2. Access by Other Users	10
4.1.3.3. Negotiable Data Protocol	11
4.1.3.4. Separation of Data Operations from Application Control	11
4.1.4. Data Management Requirements	12
4.1.4.1. Agnostic of reliability	12
4.1.4.2. Time-to-live for Stored Data	12
4.1.4.3. Offline Usage	12
4.1.5. Resource Control	12
4.1.5.1. Multiple Applications	12
4.1.5.2. Per-Peer, Per-Data Control	13
4.1.5.3. Server Involvement	13
4.1.6. Authorization	14
4.1.6.1. Per-Peer, Per-Data Read Access	14
4.1.6.2. Per-User Write Access	14
4.1.6.3. Authorization Checks	15
4.1.6.4. Credentials Not IP-Based	15
4.1.6.5. Server Involvement	15
5. Storage Requirements	15
5.1. Requirements	15
5.1.1. Explicit Deletion of Stored Data	15
5.1.2. Multiple writing	16
5.1.3. Multiple reading	16
5.1.4. Reading before completely written	16

5.1.5. Writing model	16
5.1.6. Storage Status	17
5.2. Non-Requirements	17
5.2.1. No ability to update	17
6. Implementation Considerations	17
6.1. Resource Scheduling	17
6.2. Removal of Duplicate Records	18
7. Discussion and Open Issues	18
7.1. Discussion	18
7.2. Open Issues	18
8. Security Considerations	19
9. IANA Considerations	19
10. References	19
10.1. Normative References	19
10.2. Informative References	19
Appendix A. Acknowledgments	20
Authors' Addresses	20

1. Introduction

The object of DECOupled Application Data Enroute (DECADE) is to provide an open and standard in-network storage system for applications, primarily applications that could be implemented using a content distribution paradigm, where data is broken in to one or more chunks and then distributed. This may already include many types of applications including P2P applications, IPTV, and VoD. Instead of always transferring data directly from a source-owner peer to a requesting peer, the source-owner peer can store and manage its content on its in-network storage. The requesting peer can get the address of the in-network storage pertaining to the source-owner peer and retrieve data from the storage.

This draft enumerates and explains the rationale behind SPECIFIC requirements on the protocol design and on any data store implementation that may be used to implement DECADE servers that should be considered during the design and implementation of a DECADE system. As such, it DOES NOT include general guiding principals. General design considerations, explanation of the problem being addressed, and enumeration of the types of applications to which DECADE may be suited is not considered in this document. For general information, please see the problem statement [I-D.ietf-decade-problem-statement] and architecture drafts.

This document enumerates the requirements to enable target applications to utilize in-network storage. In this context, using storage resources includes not only basic capabilities such as storing and retrieving data, and managing data, but also (1) controlling access by peers with which it is sharing data and (2) controlling the resources used by remote peers when accessing data.

This document will be updated to track revisions to the problem statement.

Editors Note: Currently the Architecture document is a WG milestone, but not yet a WG item. We have made the assumption that there will be a WG item meeting this milestone going forward.

2. Terminology and Concepts

This document uses terms defined in [I-D.ietf-decade-problem-statement]. In particular, IAP refers to the In-network storage Access Protocol, which is the protocol used to communicate between a DECADE client and DECADE server (in-network storage) for access control and resource control.

This document also defines additional terminology:

Target Application: An application (typically installed at end-hosts) with the ability to explicitly control usage of network and/or storage resources to deliver contents to a large number of users. Such applications distribute large contents (e.g., a large file, or video stream) by dividing the contents into smaller blocks for more flexible distribution (e.g., multipath). The distributed content is typically immutable (though it may be deleted). We use the term Target Application to refer to the type of applications that are explicitly (but not exclusively) supported by DECADE.

3. Requirements Structure

The DECADE protocol is intended to sit between P2P applications and a back-end storage system. In the development of DECADE, it must be made clear that the intention is to NOT develop yet another storage system, but rather to create a protocol that enables P2P applications to make use of storage within the network, leaving specific storage system considerations to the implementation of the DECADE servers as much as possible. For this reason, we have divided the requirements into three categories:

- o **General Principles:** Overall requirements that a DECADE system must conform to.
- o **Protocol Requirements:** Protocol requirements for Target Applications to make use of in-network storage within their own data dissemination schemes. Development of these requirements is guided by a study of data access, search and management capabilities used by Target Applications.
- o **Storage Requirements:** Functional requirements necessary for the back-end storage system employed by the DECADE server. Development of these requirements is guided by a study of the data access patterns used by Target Applications.

It should also be made clear that the approach is to make DECADE a simple protocol, while still enabling its usage within many P2P applications. For this reason, and to further reinforce the distinction between DECADE and a storage system, in some cases we also highlight the non-requirements of the protocol. These non-requirements are intended to capture behaviors that will NOT be assumed to be needed by DECADE's Target Applications and hence not present in the DECADE protocol.

Finally, some implementation considerations are provided, which while

strictly are not requirements, are intended to provide guidance and highlight potential points of concern that need to be considered by the protocol developers, and later by implementors.

4. Protocol Requirements

4.1. Requirements

4.1.1. Overall Protocol Requirements

4.1.1.1. Application-independent API

REQUIREMENT(S): The DECADE IAP MUST provide a simple, application-independent API for P2P applications to access in-network storage.

RATIONALE: Since the majority of existing P2P application APIs don't support in-network storage management, new application-independent API must be introduced. The API should be simple to encourage adoption, as well as to ensure that a minimum set of functions, and not an entire network storage system is implemented.

4.1.1.2. Cross-platform Access

REQUIREMENT(S): If DECADE supports the ability to store metadata associated with data objects, the DECADE protocol(s) MUST transmit any metadata using an operating system-independent and architecture-independent format.

RATIONALE: If DECADE supports the possibility for storing metadata (e.g., a description, uploaded date, or object size), a possible use for the metadata is to help a DECADE client locate a desired data object. Data objects may be stored by DECADE clients running on various platforms. To enable metadata to be readable regardless of its source it must be transmitted to and from the DECADE server in a standard format.

4.1.1.3. Connectivity Concerns

4.1.1.3.1. NATs and Firewalls

REQUIREMENT(S): DECADE SHOULD be usable across firewalls and NATs without requiring additional network support (e.g., Application-level Gateways).

RATIONALE: Firewalls and NATs are widely used in the Internet today, both in ISP networks and within households. Deployment of DECADE must not require modifications to such devices (beyond, perhaps, reconfiguration).

4.1.1.3.2. Connections to Clients

REQUIREMENT(S): DECADE SHOULD NOT require that network connections be made to DECADE clients (e.g., from a server to a DECADE client or from a DECADE client to another DECADE client).

RATIONALE: Many household networks and operating systems have firewalls and NATs configured by default. To ease deployment by avoiding configuration changes and help mitigate security risks, DECADE should not require clients to listen for any incoming network connections (beyond what is required by any other already-deployed application).

4.1.1.4. Error and Failure Conditions

4.1.1.4.1. Overload Condition

REQUIREMENT(S): In-network storage, which is operating close to its capacity limit (e.g., too busy servicing other requests), MUST be able to reject requests.

RATIONALE: When in-network storage is operating at a limit where it may not be able to process additional requests, it should not be required to generate responses to such additional requests. Forcing the in-network storage to do so can impair its ability to service existing requests.

4.1.1.4.2. Insufficient Resources

REQUIREMENT(S): DECADE MUST support an error condition indicating to a DECADE client that resources (e.g., storage space) were not available to service a request (e.g., storage quota exceeded when attempting to store data).

RATIONALE: The currently-used resource levels within the in-network storage are not locally-discoverable, since the resources (disk, network interfaces, etc) are not directly attached. In order to allocate resources appropriately amongst peers, a client must be able to determine when resource limits have been reached. The client can then respond by explicitly freeing necessary resources or waiting for such resources to be freed.

4.1.1.4.3. Unavailable and Deleted Data

REQUIREMENT(S): DECADE MUST support error conditions indicating that (1) data was rejected from being stored, (2) deleted, or (3) marked unavailable by a storage provider.

RATIONALE: Storage providers may require the ability to (1) avoid storing, (2) delete, or (3) quarantine certain data that has been identified as illegal (or otherwise prohibited). DECADE does not indicate how such data is identified, but applications using DECADE should not break if a storage provider is obligated to enforce such policies. Appropriate error conditions should be indicated to applications.

4.1.2. Transfer and Latency Requirements

4.1.2.1. Low-Latency Access

REQUIREMENT(S): DECADE SHOULD provide "low-latency" access for application clients. DECADE MUST allow clients to specify at least two classes of services for service: lowest possible latency and latency non-critical.

RATIONALE: Some applications may have requirements on delivery time (e.g., live streaming). The user experience may be unsatisfactory if the use of in-network storage results in lower performance than connecting directly to peers over a low-speed, possibly congested uplink. Additionally, the overhead required for control-plane operations in DECADE must not cause the latency to be higher than for a low-speed, possibly congested uplink. While it is impossible to make a guarantee that a system using in-network storage will always outperform a system that does not for every transfer, the overall performance of the system should be improved compared with direct connections, even considering control overhead.

4.1.2.2. Indirect Transfer

REQUIREMENT(S): DECADE MUST allow a user's in-network storage to directly fetch from other user's in-network storage.

RATIONALE: As an example, a requesting peer may get the address of the storage pertaining to the source-owner peer and then tell its own in-network storage to fetch the content from the source-owner's in-network storage. This helps to avoid extra transfers across ISP network links where possible.

4.1.2.3. Data Object Size

REQUIREMENT(S): DECADE MUST allow for efficient data transfer of small objects (e.g., 16KB) between a DECADE client and in-network storage with minimal additional latency required by the protocol.

RATIONALE: Though P2P applications are frequently used to share large amounts of data (e.g., continuous streams or large files), the data itself is typically subdivided into smaller chunks that are transferred between peers. Additionally, the small chunks may have requirements on delivery time (e.g., in a live-streaming application). DECADE must enable data to be efficiently transferred amongst peers at this granularity.

4.1.2.4. Communication among In-network Storage Elements

REQUIREMENT(S): DECADE SHOULD support the ability for two in-network storage elements in different administrative domains to store and/or retrieve data directly between each other. If such a capability is supported, this MAY be the same (or a subset or extension of) as the IAP used by clients to access data.

RATIONALE: Allowing server-to-server communication can reduce latency in some common scenarios. Consider a scenario when a DECADE client is downloading data into its own storage from another client's in-network storage. One possibility is for the client to first download the data itself, and then upload it to its own storage. However, this causes unnecessary latency and network traffic. Allowing the data to be downloaded from the remote in-network storage into the client's own in-network storage can alleviate both.

4.1.3. Data Access Requirements

4.1.3.1. Reading/Writing Own Storage

REQUIREMENT(S): DECADE MUST support the ability for a DECADE client to read data from and write data to its own in-network storage.

RATIONALE: Two basic capabilities for any storage system are reading and writing data. A DECADE client can read data from and write data to in-network storage space that it owns.

4.1.3.2. Access by Other Users

REQUIREMENT(S): DECADE MUST support the ability for a user to apply access control policies to users other than itself for its storage. The users with whom access is being shared can be under a different administrative domain than the user who owns the in-network storage. The authorized users may read from or write to the user's storage.

RATIONALE: Peers in a P2P application may be located across multiple ISPs under multiple administrative domains. Thus, to be useful by P2P applications, DECADE allows a user to specify access control policies for users that may or may not be known to the user's storage provider.

4.1.3.3. Negotiable Data Protocol

REQUIREMENT(S): DECADE MUST support the ability for a DECADE client to negotiate with its In-network storage about which protocol it can use to read data from and write data to its In-network storage.

RATIONALE: Since typical data transport protocols (e.g., NFS and WebDAV) already provide read and write operations for network storage, it may not be necessary for DECADE to define such operations in a new protocol. However, because of the particular application requirements and deployment considerations, different applications may support different protocols. Thus, a DECADE client must be able to select an appropriate protocol also supported by the in-network storage. This requirement also follows as a result of the requirement of Separation of Control and Data Operations (Section 4.1.3.4).

4.1.3.4. Separation of Data Operations from Application Control

REQUIREMENT(S): The DECADE IAP MUST only provide a minimal set of core operations to support diverse policies implemented and desired by Target Applications.

RATIONALE: Target Applications support many complex behaviors and diverse policies to control and distribute data, such as (e.g., search, index, setting permissions/passing authorization tokens). Thus, to support such Target Applications, these behaviors must be logically separated from the data transfer operations (e.g., retrieve, store). Some minimal overlap (for example obtaining credentials needed to encrypt or authorize data transfer using control operations) may be required to be directly supported by DECADE.

4.1.4. Data Management Requirements

4.1.4.1. Agnostic of reliability

REQUIREMENT(S): DECADE SHOULD remain agnostic of reliability/fault-tolerance level offered by storage provider.

RATIONALE: Providers of a DECADE service may wish to offer varying levels of service for different applications/users. However, a single compliant DECADE client should be able to use multiple DECADE services with differing levels of service.

4.1.4.2. Time-to-live for Stored Data

REQUIREMENT(S): DECADE MUST support the ability for a DECADE client to indicate a time-to-live value (or expiration time) indicating a length of time until particular data can be deleted by the in-network storage element.

RATIONALE: Some data stored by a DECADE client may be usable only within a certain window of time, such as in live-streaming P2P applications. Providing a time-to-live value for stored data (e.g., at the time it is stored) can reduce management overhead by avoiding many 'delete' commands sent to in-network storage. The in-network storage may still keep the data in cache for bandwidth optimization. But this is guided by the privacy policy of the DECADE provider.

4.1.4.3. Offline Usage

REQUIREMENT(S): DECADE MAY support the ability for a user to provide authorized access to its in-network storage even if the user has no DECADE applications actively running or connected to the network.

RATIONALE: If an application desires, it can authorize peers to access its storage even after the application exits or network connectivity is lost. An example use case is mobile scenarios, where a client can lose and regain network connectivity very often.

4.1.5. Resource Control

4.1.5.1. Multiple Applications

REQUIREMENT(S): DECADE SHOULD support the ability for users to define resource sharing policies for multiple applications being run/managed by the user.

RATIONALE: A user may own in-network storage and share the in-network storage resources amongst multiple applications. For example, the user may run a video-on-demand application and a live-streaming (or even two different live-streaming applications) application which both make use of the user's in-network storage. The applications may be running on different machines and may not directly communicate. Thus, DECADE should enable the user to determine resource sharing policies between the applications.

One possibility is for a user to indicate the particular resource sharing policies between applications out-of-band (not using a standard protocol), but this requirement may manifest itself in passing values over IAP to identify individual applications. Such identifiers can be either user-generated or server-generated and do not need to be registered by IANA.

4.1.5.2. Per-Peer, Per-Data Control

REQUIREMENT(S): A DECADE client MUST be able to assign resource quotas to individual peers for reading from and writing particular data to its in-network storage within a particular range of time. The DECADE server MUST enforce these constraints.

RATIONALE: P2P applications can rely on control of resources on a per-peer or per-data basis. For example, application policy may indicate that certain peers have a higher bandwidth share for receiving data. Additionally, certain data (e.g., chunks) may be distributed with a higher priority. As another example, when allowing a remote peer to write data to a user's in-network storage, the remote peer may be restricted to write only a certain amount of data. Since the client may need to manage multiple peers accessing its data, it should be able to indicate the time over which the granted resources are usable. For example, an expiration time for the access could be indicated to the server after which no resources are granted (e.g., indicate error as access denied).

4.1.5.3. Server Involvement

REQUIREMENT(S): A DECADE client MUST be able to indicate, without contacting the server itself, resource control policies for peers' requests.

RATIONALE: One important consideration for in-network storage elements is scalability, since a single storage element may be used to support many users. Many P2P applications use small chunk sizes and frequent data exchanges. If such an application employed resource control and contacted the in-network storage element for each data exchange, this could present a scalability challenge for the server as well as additional latency for clients.

An alternative is to let requesting users get the resource control policies and users can then present the policy to the storage directly. This can result in reduced messaging handled by the in-network storage.

4.1.6. Authorization

4.1.6.1. Per-Peer, Per-Data Read Access

REQUIREMENT(S): A DECADE Client MUST be able to authorize individual peers to read particular data stored on its in-network storage.

RATIONALE: A P2P application can control certain application-level policies by sending particular data (e.g., chunks) to certain peers. It is important that peers not be able to circumvent such decisions by arbitrarily reading any currently-stored data in in-network storage.

4.1.6.2. Per-User Write Access

REQUIREMENT(S): A DECADE Client MUST be able to authorize individual peers to store data into its in-network storage.

RATIONALE: The space managed by a user in in-network storage can be a limited resource. At the same time, it can be useful to allow remote peers to write data directly to a user's in-network storage. Thus, a DECADE client should be able to grant only certain peers this privilege.

Note that it is not (currently) a requirement to check that a peer stores a particular set of data (e.g., the check that a remote peer writes the expected chunk of a file). Enforcing this as a requirement would require a client to know which data is expected (e.g., the full chunk itself or a hash of the chunk) which may not be available in all applications. Checking for a

particular hash could be considered as a requirement in the future that could optionally be employed by applications.

4.1.6.3. Authorization Checks

REQUIREMENT(S): In-network storage MUST check the authorization of a client before it executes a supplied request. The in-network storage MAY use optimizations to avoid such authorization checks as long as the enforced permissions are the the same.

RATIONALE: Authorization granted by a DECADE client are meaningless unless unauthorized requests are denied access. Thus, the in-network storage element must verify the authorization of a particular request before it is executed.

4.1.6.4. Credentials Not IP-Based

REQUIREMENT(S): Access MUST be able to be granted on other credentials than the IP address

RATIONALE: DECADE clients may be operating on hosts without constant network connectivity or without a permanent attachment address (e.g., mobile devices). To support access control with such hosts, DECADE servers must support access control policies that use information other than IP addresses.

4.1.6.5. Server Involvement

REQUIREMENT(S): A DECADE client MUST be able to indicate, without contacting the server itself, access control policies for peers' requests.

RATIONALE: See discussion in Section 4.1.5.3.

5. Storage Requirements

5.1. Requirements

5.1.1. Explicit Deletion of Stored Data

REQUIREMENT(S): DECADE MUST support the ability for a DECADE client to explicitly delete data from its own in-network storage.

RATIONALE: A DECADE client may continually be writing data to its in-network storage. Since there may be a limit (e.g., imposed by the storage provider) to how much total storage can be used, some data may need to be removed to make room for additional data. A

DECADE client should be able to explicitly remove particular data. This may be implemented using existing protocols.

5.1.2. Multiple writing

REQUIREMENT(S): DECADE MUST NOT allow multiple writers for the same object. Implementations raise an error to one of the writers.

RATIONALE: This avoids data corruption in a simple way while remaining efficient.

5.1.3. Multiple reading

REQUIREMENT(S): DECADE MUST allow for multiple readers for an object..

RATIONALE: One characteristic of P2P applications is the ability to upload an object to multiple peers. Thus, it is natural for DECADE to allow multiple readers to access the content concurrently.

5.1.4. Reading before completely written

REQUIREMENT(S): DECADE MAY allow readers to read from objects before they have been completely written.

RATIONALE: Some P2P systems (in particular, streaming) can be sensitive to latency. A technique to reduce latency is to remove store-and-forward delays for data objects (e.g., make the object available before it is completely stored). Appropriate handling for error conditions (e.g., a disappearing writer) needs to be specified.

5.1.5. Writing model

REQUIREMENT(S): DECADE MUST provide at least a writing model (while storing an object) that appends data to data already stored.

RATIONALE: Depending on the object size (e.g., chunk size) used by a P2P application, the application may need to send data to the DECADE server in multiple packets. To keep implementation simple, the DECADE must at least support the ability to write the data sequentially in the order received. Implementations MAY allow application to write data in an object out-of-order (but MAY NOT overwrite ranges of the object that have already been stored).

5.1.6. Storage Status

REQUIREMENT(S): A DECADE client MUST be able to retrieve current resource usage (including list of stored data) and resource quotas on its in-network storage.

RATIONALE: The resources used by a client are not directly-attached (e.g., disk, network interface, etc). Thus, the client cannot locally determine how such resources are being used. Before storing and retrieving data, a client should be able to determine which data is available (e.g., after an application restart). Additionally, a client should be able to determine resource availability to better allocate them to remote peers.

5.2. Non-Requirements

5.2.1. No ability to update

REQUIREMENT(S): DECADE SHOULD NOT provide ability to update existing objects. That is, objects are immutable once they are stored.

RATIONALE: Reasonable consistency models for updating existing objects would significantly complicate implementation (especially if implementation chooses to replicate data across multiple servers). If a user needs to update a resource, it can store a new resource and then distribute the new resource instead of the old one.

6. Implementation Considerations

The intent of this section is to collect discussion items and implementation considerations that have been discovered as this requirements document has been produced. The content of this section will be migrated to an appropriate place as the document and the Working Group progress.

6.1. Resource Scheduling

The particular resource scheduling policy may have important ramifications on the performance of applications. This document has explicitly mentioned simultaneous support for both low-latency applications and latency-tolerant applications.

Denial of Service attacks may be another risk. For example, rejecting new requests due to overload conditions may introduce the ability to perform a denial of service attack depending on a particular DECADE server's scheduling implementation and resource

allocation policies.

6.2. Removal of Duplicate Records

There are actually two possible scenarios here. The first is the case of removing duplicates within one particular DECADE server (or logical server). While not a requirement, as it does not impact the protocol and is technically not noticeable on message across the wire, a DECADE server may implement internal mechanisms to monitor for duplicate records and use internal mechanisms to prevent duplication of internal storage.

The second scenario is removing duplicates across a distributed set of DECADE servers. This is a more difficult problem, and if the group decides to support this capability, it may require protocol support. See Section 7.2 for more details.

7. Discussion and Open Issues

7.1. Discussion

Sometimes, several logical in-network storages could be deployed on the same physical network device. In this case, in-network storages on the same physical network device can communicate and transfer data through internal communication messages. However in-network storages deployed on different physical network devices **SHOULD** communicate with in-network storage Access Protocol (IAP).

To provide fairness among users, the in-network storage provider should assign resource (e.g., storage, bandwidth, connections) quota for users. This can prevent a small number of clients from occupying large amounts of resources on the in-network storage, while others starve.

7.2. Open Issues

Gaming of the Resource Control Mechanism: There has been some discussion of how applications may be able game the scheduling system by manipulating the resource control mechanism, for example by specifying many small peers to increase total throughput. This is a serious concern, and we need to identify specific requirements on the protocol (hopefully independent of particular scheduling/resource control schemes) to help address this.

Discovery: There needs to be some mechanism for a user to discover that there is a DECADE service available for their use, and to locate that server. This is particularly important in the case of mobile applications, since the actual servers that are available at any given time may differ. However, the specifics of what mechanisms (DHCP, HTTP page, etc.) have not been discussed, or even if the protocol should specify one or leave it as an implementation detail. This needs to be defined, and specific requirements formulated if needed.

Removal of Duplicate Records Across Servers: If the group wishes to allow for automated mechanisms to remove duplicates across a number of separate servers, some protocol support may need to be added. In the case of removing duplicates within a single (logical) DECADE server, this is simply an implementation concern. See Section 6 for more details.

8. Security Considerations

Authorization for access to in-network storage is an important part of the requirements listed in this document. Authorization for access to storage resources and the data itself is important for users to be able to manage and limit distribution of content. For example, a user may only wish to share particular content with certain peers.

If the authorization technique implemented in DECADE passes any private information (e.g., user passwords) over the wire, it MUST be passed in a secure way.

9. IANA Considerations

There are no IANA considerations with this document.

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-decade-problem-statement]
Yongchao, S., Zong, N., Yang, Y., and R. Alimi, "DECoupled

Application Data Enroute (DECADE) Problem Statement",
draft-ietf-decade-problem-statement-00 (work in progress),
August 2010.

[LLSB08] Dave Levin, Katrina LaCurts, Neil Spring, Bobby
Bhattacharjee., "BitTorrent is an Auction: Analyzing and
Improving BitTorrent's Incentives", In SIGCOMM 2008.

[PPLive] "PPLive", <http://www.pplive.com>.

Appendix A. Acknowledgments

We would also like to thank Haibin Song for substantial contributions to earlier versions of this document. We would also like to thank Reinaldo Penno, Alexey Melnikov, Rich Woundy, Ning Zong, Roni Even, David McDysan, Boerje Ohlman and Dirk Kutscher for contributions (including some text used verbatim) and general feedback.

Authors' Addresses

Yingjie Gu
Huawei
No. 101 Software Avenue
Nanjing, Jiangsu Province 210012
P.R.China

Phone: +86-25-56624760
Email: guyingjie@huawei.com

David A. Bryan
Cogent Force, LLC / Huawei

Email: dbryan@ethernet.org

Yang Richard Yang
Yale University

Email: yry@cs.yale.edu

Richard Alimi
Google

Email: ralimi@google.com

DECADE
Internet-Draft
Intended status: Informational
Expires: April 25, 2011

R. Alimi, Ed.
Yale University
A. Rahman, Ed.
InterDigital Communications, LLC
Y. Yang, Ed.
Yale University
October 22, 2010

A Survey of In-network Storage Systems
draft-ietf-decade-survey-01

Abstract

This document surveys deployed and experimental in-network storage systems and describes their applicability for DECADE.

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 April 25, 2011.

Copyright Notice

Copyright (c) 2010 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. Survey Overview	3
2.1. Terminology and Concepts	3
2.2. Historical Context	3
3. In-network Storage System Components	5
3.1. Data Access Interface	5
3.2. Data Management Operations	5
3.3. Data Search Capability	5
3.4. Access Control Authorization	5
3.5. Resource Control Interface	6
3.6. Discovery Mechanism	6
3.7. Storage Mode	6
4. In-Network Storage Systems	6
4.1. Amazon S3	6
4.2. BranchCache	8
4.3. Cache-and-Forward Architecture	10
4.4. CDN	11
4.5. Delay-Tolerant Network	13
4.6. Named Data Networking	14
4.7. Network Traffic Redundancy Elimination	16
4.8. OceanStore	17
4.9. Photo Sharing	18
4.10. P2P Cache	19
4.11. Web Cache	22
4.12. Usenet	23
4.13. Observations Regarding In-Network Storage Systems	25
5. Storage And Other Related Protocols	25
5.1. iSCSI	25
5.2. HTTP	26
5.3. NFS	28
5.4. OAuth	29
5.5. WebDAV	30
5.6. Observations Regarding Storage and Related Protocols	32
6. Conclusions	33
7. Security Considerations	33
8. IANA Considerations	33
9. Acknowledgments	33
10. Revision History	33
11. Informative References	34
Appendix A. Authors	37
Authors' Addresses	37

1. Introduction

DECADE (DECoupled Application Data Enroute) is an architecture that provides applications with access to in-network storage.

A major motivation for DECADE is the substantial increase on capacity and reduction in cost offered by storage systems. In particular, over the last decade, capacity of solid-state storage has increased 100-fold, while cost dropped to \$50/GB; capacity of magnetic storage devices has increased 100-fold, while cost dropped to \$0.50/GB.

High-capacity and low-cost in-network storage devices introduce substantial opportunities. One example of in-network storage is content caches supporting Web and P2P content. Different from existing content caches whose control fully reside at the owners of the caching devices, DECADE also allows applications to control access to their allocated in-network storage, as well as the resources consumed while accessing that storage (bandwidth, connections, storage space). While designed in the context of P2P applications, it may be useful to other applications as well. This document provides details on deployed and experimental in-network storage solutions, and evaluates their suitability for DECADE.

We note that the techniques presented in this section are only representative of the research in this area. Rather than trying to enumerate an exhaustive list, we have chosen some typical techniques that lead to derivative works.

2. Survey Overview

2.1. Terminology and Concepts

This document uses terms defined in [1].

2.2. Historical Context

In-network storage has been used previously in numerous scenarios to reducing network traffic and enable more efficient content distribution. This section presents a brief history of content distribution techniques and illustrates how DECADE relates to past approaches. Systems have been developed with particular use cases in mind. Thus, this survey is not meant to point out shortcomings of existing solutions, but rather to indicate where certain capabilities required in DECADE [2] are not provided by existing systems.

In the early stage of Internet development, most Web content was stored at a central server and clients requested Web content from the

central server. In this architecture, the central server was required to provide a large amount of bandwidth. Web browsing is still a primary activity on today's Internet. As more and more users access Web content, a central server can become overloaded. The use of web caches is one technique to reduce load on a central server. Web caches store frequently-requested content, and provide bandwidth for serving the content to clients.

The ongoing growth of broadband technology in the worldwide market has been driven by the hunger of customers for new multimedia services as well as Web content. In particular, the use of audio and video streaming formats has become common for delivery of rich information to the public - both residential and business.

To overcome this challenge of massive multimedia consumption, only installing more Web cache will not be enough. Moving content closer to the consumer results in greater network efficiency, improved QoS, and lower latency, while facilitating personalization of content through broadband content applications. In these edge technologies, CDN is a representative technique. Content Delivery Networks (CDN) is based on a large-scale distributed network of servers located closer to the edges of the Internet for efficient delivery of digital content including various forms of multimedia content.

Although CDN is an effective means of information access and delivery, there are two barriers to making CDN a more common service: cost and replication integrity. Deploying a CDN for publicly available content is expensive. It requires administrative control over nodes with large storage capacity at geographically dispersed locations with adequate connectivity. CDN can be scalable, but due to this administrative and cost overhead, not rapidly deployable for the common user.

The emergence and maturity of Peer to Peer (P2P) has allowed improvements to many network applications. P2P allows the use of client resources, such as CPU, memory, storage, and bandwidth, for serving content. This can reduce the amount of resources required by a content provider. Multimedia content delivery using various peer-to-peer or peer-assisted frameworks has been shown to greatly reduce the dependence on CDN and central content servers. However, popularity of P2P applications has resulted in increased traffic on ISP networks.

DECADE aims to provide a standard protocol allowing P2P applications (including Content Providers) to make use of in-network storage to reduce the traffic burden on ISP networks, while enabling P2P applications to control access to content they have placed in in-network storage.

3. In-network Storage System Components

Before surveying individual technologies, we describe the basic components of in-network storage systems used to evaluate them in the context of DECADE. For consistency and to easy comparison, we use the same model to evaluate each storage technology in this document.

Note that the network protocol(s) used by a storage system are also an important part of the design. We omit details of particular protocol choices in the current version of this document.

3.1. Data Access Interface

A set of operations are available to a user for accessing data in the in-network storage. Solutions typically allow both read and write, though the mechanisms for doing so can differ drastically.

3.2. Data Management Operations

Storage systems may provide users the ability to manage stored content. For example, operations such as delete and move can be provided to users. In this survey, we focus on data management operations that are provided to client users and omit those provided to system administrators.

3.3. Data Search Capability

Some storage systems may provide the capability to search or enumerate content that has been stored. In this survey, we focus on search capabilities that are provided to client users and omit those provided to system administrators.

3.4. Access Control Authorization

A user or some other entity defines the access policies for the in-network storage. The in-network storage system then checks the authorization of a client before it stores or retrieves content. We define three types of access control authorization: public-unrestricted, public-restricted, and private.

Public-unrestricted refers to content on an in-network storage system that is widely available to everyone (i.e. without restrictions). An analogy is accessing www.Wikipedia.org on the Internet, or anonymous access to FTP sites.

Public-restricted refers to content on an in-network storage system that is generally available to the public but which is restricted by certain criteria (e.g. country or network), but which does not

require any confidential credentials from the client. An analogy is some content (e.g. on line TV show episode) on the Internet that can only be viewable in selected countries (i.e. white/black lists or black-outs).

Private refers to content on an in-network storage system that is only made available to one or more clients presenting the required confidential credentials (e.g. password or key). This content is not available to anyone without the proper confidential access credentials.

3.5. Resource Control Interface

This is the interface through which users manage the resources on in-network storage that can be used by other peers, e.g., the bandwidth or connections. The storage system may also allow users to indicate a time for which resources are granted.

3.6. Discovery Mechanism

Users use the discovery mechanism to find location of in-network storage, find access interface or resource control interface or other interfaces of in-network storage.

3.7. Storage Mode

The data managed by the in-network storage could be of various types. Example storage modes are file-based, object-based, or block-based.

4. In-Network Storage Systems

This section surveys in-network storage systems using the methodology defined above. The survey includes some systems that are widely deployed today, some systems that are just being deployed, and some experimental/futuristic systems. The survey covers both traditional client-server architectures and P2P architectures. The surveyed systems are listed in alphabetical order. Also, for each system, a brief explanation is given of the relevance to DECADE.

4.1. Amazon S3

Amazon S3 (Simple Storage Service) [19] provides an online storage service using web (HTTP) interfaces. Users create buckets, and each bucket can contain stored objects. Users are provided an interface through which they can manage their buckets. Amazon S3 is popular backend storage for other services. Other related storage services is the Blob Service provided by Windows Azure [20], and the Google

Storage for Developers [21].

4.1.1. Applicability to DECADE

Very widely used (deployed) example of in-network storage. Amazon leases the storage to third party companies for disparate services. In particular, Amazon S3 has a rich model for authorization (using signed queries) to integrate with a wide variety of use cases. A focus for Amazon S3 is scalability. Particular simplifications that were made are the absence of a general, hierarchical namespace and the inability to update contents of existing data.

4.1.2. Data Access Interface

Users can read, and write objects.

4.1.3. Data Management Operations

Users can delete previously-stored objects.

4.1.4. Data Search Capability

Users can list contents of buckets to find objects matching desired criteria.

4.1.5. Access Control Authorization

All methods of access control are supported: public-unrestricted, public-restricted and private.

For example, access to stored objects can be restricted by owner, a list of other Amazon Web Service users, all Amazon Web Service Users, or open to all users (anonymous access). Another option is for the owner to generate and sign a query (e.g., a query to read an object) that can be used by any user until an owner-defined expiration time.

4.1.6. Resource Control Interface

Not provided.

4.1.7. Discovery Mechanism

Users are provided a well-known DNS name (either a default provided by Amazon, or one customized by a particular user). Users accessing S3 storage use DNS to discover an IP address where S3 requests can be sent.

4.1.8. Storage Mode

Object-based, with the extension that objects can be organized into user-defined buckets.

4.2. BranchCache

BranchCache [25] is a feature integrated into Windows (Windows 7 and Windows Server 2008R2) that aims to optimize enterprise branch office file access over the WAN links. The main goals are to reduce WAN link utilization and improve application responsiveness by caching and sharing content within a branch while still maintaining end-to-end security. BranchCache allows files retrieved from the web servers and file servers located in headquarters or datacenters to be cached in remote branch offices, and shared among users in the same branch accessing the same content. BranchCache operates transparently by instrumenting the HTTP and SMB components of the networking stack. It provides two modes of operation: Distributed Cache and Hosted Cache.

In both modes, a client always contacts a BranchCache-enabled content server first to get the content identifiers for local search. If the content is cached locally, the client then retrieves the content within the branch. Otherwise, the client will go back to the original content server to request the content. The two modes differ in how the content is shared.

In the Hosted Cache mode, a locally provisioned server acts as a cache for files retrieved from the servers. After getting the content identifiers, the client first consults the cache for the desired file. If it is not present in the cache, the client retrieves it from the content server and sends it to the cache for storage.

In the Distributed Cache mode, a client first queries other clients in the same network using the Web Services Discovery multicast protocol. As in the Hosted Cache mode, the client retrieves the file from the content server it is not available locally. After retrieving the file (either from another client or the content server), the client stores the file locally.

The original content server always authorizes requests from clients. Cached content is encrypted, and clients can only decrypt the data using keys derived from metadata returned by the content server. In addition to instrumenting the networking stack at clients, content servers must also support BranchCache.

4.2.1. Applicability to DECADE

BranchCache is an example of an in-network storage system primarily targeted at enterprise networks. It supports both a P2P like mode (Distributed Cache) as well as a client-server mode (Hosted Cache). Integration into the Microsoft OS will ensure wide distribution of this in-network storage technology.

4.2.2. Data Access Interface

Clients transparently retrieve (read) data from a cache (other clients or a Hosted Cache) since it operates by instrumenting the networking stack. In Hosted Cache mode, clients write data to the Hosted Cache once it is retrieved from the content server.

4.2.3. Data Management Operations

Not provided.

4.2.4. Data Search Capability

Not provided.

4.2.5. Access Control Authorization

Access control method is private. For example, transferred content is encrypted, and can only be decrypted by keys derived from data received from the original content server. Though data may be transferred to unauthorized clients, end-to-end security is maintained by only allowing authorized clients to decrypt the data.

4.2.6. Resource Control Interface

The storage capacity of caches on the clients and Hosted Caches are configurable by system administrators. The Hosted Cache further allows configuration of the maximum number of simultaneous client accesses. In the Distributed Caching mode, exponential back-off and throttling mechanisms are utilized to prevent reply storms of popular content requests. The client will also spread data block access among multiple serving clients that have the content (complete or partial) to improve latency and provide some load balancing.

4.2.7. Discovery Mechanism

The Distributed Cache mode uses multicast for discovery of other clients and content within a local network. Currently, the Hosted Cache mode uses policy provisioning or manual configuration of the server used as the Hosted Cache.

4.2.8. Storage Mode

Object-based.

4.3. Cache-and-Forward Architecture

Cache-and-Forward (CNF) [24] is an architecture for content delivery services for the future Internet. In this architecture, storage can be exploited at nodes within the network, either directly at routers or deployed nearby the routers. CNF is based on the concept of store-and-forward routers with large storage, providing for opportunistic delivery to occasionally disconnected mobile users and for in-network caching of content. The proposed CNF protocol uses reliable hop-by-hop transfer of large data files between CNF routers in place of an end-to-end transport protocol like TCP.

4.3.1. Applicability to DECADE

An example of an experimental in-network storage system that would require storage space on (or near) a large number of routers in the Internet if it was deployed. As the name of the system implies, it would provide short term caching and not long term network storage.

4.3.2. Data Access Interface

Users implicitly store content at Cache-and-forward routers by requesting files. End hosts read content from in-network storage by submitting queries for content.

4.3.3. Data Management Operations

Not provided.

4.3.4. Data Search Capability

Not provided.

4.3.5. Access Control Authorization

Access control method is public-restricted (to any client which is part of the cache-and-forward network).

4.3.6. Resource Control Interface

Not provided.

4.3.7. Discovery Mechanism

A query including a location-independent content ID is sent to the network, and routed to a Cache-and-forward router, which handles retrieval of the data and forwarding to the end host.

4.3.8. Storage Mode

Object-based (with objects representing individual files). The architecture proposes to cache large files at storage within the network, though files could be made to represent smaller chunks of larger files.

4.4. CDN

A Content Delivery Network (CDN) provides services that improve network performance by maximizing bandwidth, improving accessibility and maintaining correctness through content replication. They offer fast and reliable applications and services by distributing content to cache or edge servers located close to users. See [18] for an additional taxonomy and survey.

A CDN has some combination of content-delivery, request-routing, distribution and accounting infrastructure. The content-delivery infrastructure consists of a set of edge servers (also called surrogates) that deliver copies of content to end-users. The request-routing infrastructure is responsible to directing client request to appropriate edge servers. It also interacts with the distribution infrastructure to keep an up-to-date view of the content stored in the CDN caches. The distribution infrastructure moves content from the origin server to the CDN edge servers and ensures consistency of content in the caches. The accounting infrastructure maintains logs of client accesses and records the usage of the CDN servers. This information is used for traffic reporting and usage-based billing.

In practice, CDN typically host static content including images, video, media clips, advertisements, and other embedded objects for dynamic Web content. A focus for CDNs is the ability to publish and deliver content to end-users in a reliable and timely manner. A CDN focuses on building its network infrastructure to provide the following services and functionalities: storage and management of content; distribution of content among surrogates; cache management; delivery of static, dynamic and streaming content; backup and disaster recovery solutions; and monitoring, performance measurement and reporting.

Examples of existing CDNs are Akamai, Limelight, and CloudFront.

The following description uses the term Content Provider to refer to the entity purchasing CDN service, and the term Client to refer to the subscriber requesting content via the CDN from the Content Provider.

4.4.1. Applicability to DECADE

Very widely used (deployed) example of in-network storage for multimedia content. The existence and operation of the storage is totally transparent to the end user. CDNs typically require a strong business relationship between the content providers and content distributors and often the business relationship extends to the ISPs.

4.4.2. Data Access Interface

CDN is typically an internal closed system, and CDN just provide read (retrieve) access interface to clients but they don't provide write (store) access interface to clients. Content provider can access to network edge servers and store content to them, or edge servers retrieve content from content provider, but client nodes just can retrieve content from edge servers.

4.4.3. Data Management Operations

Content Provider can manage the data distributed in different cache nodes, such as moving one hot data from one cache node to another cache node, or deleting one rarely-accessed data in one cache node, but client user nodes have no right to perform these operations.

4.4.4. Data Search Capability

Content provider can search or enumerate what data each cache node hold, but client user nodes have no right to perform these operations.

4.4.5. Access Control Authorization

All methods of access control are supported: public-unrestricted, public-restricted and private. Some CDN edge servers will allow usage of HTTP basic authentication with the origin server, restrictions by IP address, or they can use a token-based technique to allow the origin server to apply its own authorization criteria.

4.4.6. Resource Control Interface

Not provided.

4.4.7. Discovery Mechanism

Content providers can directly find internal CDN cache nodes to store content, since they typically have an explicit business relationship. Clients can locate CDN nodes through DNS or other redirection mechanism.

4.4.8. Storage Mode

A file-based storage mode is typically used. In most cases, CDN cache nodes cache the entire file from content provider, and sometimes they also can only cache some objects, such as file prefix or file suffix.

4.5. Delay-Tolerant Network

The Delay-Tolerant Network (DTN) [12] is an evolution of an architecture originally designed for the Interplanetary Internet. The Interplanetary Internet is a communication system envisioned to provide Internet-like services across interplanetary distances in support of deep space exploration. The DTN architecture can be utilized in various operational environments characterized by severe communication disruptions, disconnections and high-delays (e.g. a month long loss of connectivity between two planetary networks because of high solar radiation due to sun spots). The DTN architecture is thus suitable for environments including deep space networks, sensor-based networks, certain satellite networks and underwater acoustic networks.

A key aspect of the DTN is a store and forward overlay layer called the "Bundle Protocol" or "Bundle Layer" that exists between the transport and application layers [13]. The Bundle Layer forms a logical overlay that employs persistent storage to help combat long term network interruptions by providing a store and forwarding service. While traditional IP networks are also based on store and forward principles, the amount of time of a packet being kept in "storage" at a traditional IP router is typically in the order of milli-seconds (or less). In contrast, the DTN architecture assumes that most Bundle Layer nodes will use some form of persistent storage (e.g. hard disk, flash memory, etc.) for DTN packets because of the nature of the DTN environment.

4.5.1. Applicability to DECADE

An example of an experimental in-network storage system that would require fundamental changes to the Internet protocols.

4.5.2. Data Access Interface

Users implicitly cause content to be stored (until successfully forwarded) at Bundle Layer nodes by initiating/terminating any transaction that traverses the DTN.

4.5.3. Data Management Operations

Users can implicitly cause deletion of content stored at Bundle Layer nodes via a "Time To Live" type parameter that the user can control (for transactions originating from the user).

4.5.4. Data Search Capability

Not applicable.

4.5.5. Access Control Authorization

Access control method is public-restricted (to any client which is part of the DTN) or private.

4.5.6. Resource Control Interface

Not provided.

4.5.7. Discovery Mechanism

A Uniform Resource Identifier (URI) approach is used as the basis of the addressing scheme for DTN transactions (and subsequent store and forward routing through the DTN network).

4.5.8. Storage Mode

DTN applications send data to the Bundle Layer which then breaks the data into segments. These segments are then routed through the DTN network, and stored in Bundle Layer nodes as required (before being forwarded).

4.6. Named Data Networking

Named Data Networking (NDN) [33] is a research initiative which proposes to move to a new model of addressing and routing for the Internet. Specifically, NDN proposes to change Internet routing from the traditional stateless IP address based routing to a stateful routing based on "named data". Each NDN Data packet will be assigned a content name and will also be cryptographically signed. Communication is driven by the requesting end. The requester will send out an "Interest" packet which identifies the name of the data

that it wants. Routers that receive this Interest packet will remember the interface it came from and will then forward it on a named-based routing protocol. Once an Interest packet reaches a node that has the desired data, a named Data packet is sent back, which carries both the name and content of the data, along with a digital signature of the producer. This named Data packet is then forwarded back to the original requester on the reverse path of the Interest packet [34].

A key aspect of NDN is that it requires routers to store a copy of the data that it has previously routed. If a request for the same data (i.e. same name) comes to the router, then the NDN router will forward the stored data. Current communication theory is based on the capacity of a point-to-point channel (i.e. Shannon's law). The proponents of NDN believe that the capacity of the network can be extended if network storage memory has a larger and more central role in communications.

4.6.1. Applicability to DECADE

An example of an experimental in-network storage system that would require storage space on a large number of routers in the Internet. Named Data packets would be kept in storage in the NDN routers and provided to new requesters of the same data.

4.6.2. Data Access Interface

Users implicitly store content at NDN routers by requesting content (named Data packets) from the network. Subsequent requests by different users for the same content will cause the named Data packets to be read from the NDN routers in-network storage.

4.6.3. Data Management Operations

Users do not have the direct ability to delete content stored in the NDN routers. However, there will be some type of "Time To Live" parameter associated with the named Data packets though this has not yet been specified.

4.6.4. Data Search Capability

Not applicable.

4.6.5. Access Control Authorization

All methods of access control are supported: public-unrestricted, public-restricted and private.

The basic security mechanism in NDN is for the sender to digitally sign the content (named Data packet) that it sends. It is envisioned that a complete access control system can be built on top of this though this has not yet been specified.

4.6.6. Resource Control Interface

Not provided.

4.6.7. Discovery Mechanism

Names are used as the basis of the addressing and discovery scheme for NDN (and subsequent store and forward routing through the NDN network). NDN names are assumed to be hierarchical and to be able to be deterministically constructed. This is still an active area of research.

4.6.8. Storage Mode

NDN sends named Data packets through the network. These Data packets are routed through the NDN network, and stored in NDN routers.

4.7. Network Traffic Redundancy Elimination

Redundancy Elimination (RE) (e.g., [23]) is used for identifying and removing repeated content from network transfers. This technique has been proposed to improve network performance in many types of networks, such as ISP backbones and enterprise access links. One example of redundancy elimination proposal is SmartRE, proposed by Anand et al., which focuses on network-wide redundancy elimination. In packet-level redundancy elimination, forwarding elements are equipped with additional storage which can be used to cache data from forwarded packets. Upstream routers may replace packet data with a fingerprint that tells a downstream router how to decode and reconstruct the packet based on cached data.

4.7.1. Applicability to DECADE

An example of an experimental in-network storage system that would require a large amount of associated packet processing at routers if it was ever deployed.

4.7.2. Data Access Interface

Redundancy-elimination are typically transparent to the user. Writing into the storage is done by transferring data that has not already been cached. Storage is read when users transmit data identical to previously-transmitted data.

4.7.3. Data Management Operations

Not provided.

4.7.4. Data Search Capability

Not provided.

4.7.5. Access Control Authorization

Access control method is public-restricted (to any client which is part of the RE network). Note that the content provider still retains control over which peers receive the requested data. The returned data is simple "compressed" as it is transferred within the network.

4.7.6. Resource Control Interface

Not provided. The content provider still retains control over the rate at which packets are sent to a peer. The packet size within the network may be reduced.

4.7.7. Discovery Mechanism

No discovery mechanism is necessary. Routers can use redundancy-elimination without the users' knowledge.

4.7.8. Storage Mode

Object-based, with "objects" being data from packets transmitted within the network.

4.8. OceanStore

OceanStore [22] is a storage platform developed at UC Berkeley that provides globally-distributed storage. OceanStore implements a model where multiple storage providers can pool resources together. Thus, a major focus is on resiliency and self-organization and self-maintenance.

The protocol is resilient to some storage nodes being compromised by utilizing Byzantine agreement and erasure codes to store data at primary replicas.

4.8.1. Applicability to DECADE

An example of an experimental in-network storage system that provides a high degree of network resilience to failure scenarios.

4.8.2. Data Access Interface

Users may read and write objects

4.8.3. Data Management Operations

Objects may be replaced by newer versions, and multiple versions of an object may be maintained.

4.8.4. Data Search Capability

Not provided.

4.8.5. Access Control Authorization

Provided, but specifics are unclear from published paper.

4.8.6. Resource Control Interface

Not provided.

4.8.7. Discovery Mechanism

Users require an entry-point into the system in the form of one storage node that is part of OceanStore.

4.8.8. Storage Mode

Object-based, though interfaces have been provided for NFS and HTTP.

4.9. Photo Sharing

There are a growing number of popular on line Photo Sharing (storing) systems. For example, the Kodak Gallery system [28] serves over 60 million users and stores billions of images [29]. Other well known examples of Photo Sharing systems include Flickr [30] and ImageShack [31]. Also there are a number of popular blogging services, such as Tumblr [32], which specialize in also sharing large numbers of photos and other multimedia content (e.g. video, text, audio, etc.) as part of their service. All these in-network storage systems utilize both free and paid subscription models.

Most Photo Sharing systems are traditional client-server architecture. However, a minority of systems also offer a P2P mode of operation. The client-server architecture traditionally is based on HTTP with a browser client and a web server.

4.9.1. Applicability to DECADE

Very widely used (deployed) example of in-network storage where the end user has direct visibility and extensive control of the system. Typical end user interface is through a HTTP based web browser.

4.9.2. Data Access Interface

Users can read (view) and write (store) photos.

4.9.3. Data Management Operations

Users can delete previously stored photos.

4.9.4. Data Search Capability

Users can tag photos and/or organize them using sophisticated web photo album generators. Users can then search for objects (photos) matching desired criteria.

4.9.5. Access Control Authorization

Access control method is typically either private or public-unrestricted.

4.9.6. Resource Control Interface

Not provided.

4.9.7. Discovery Mechanism

Usually by manually logging on to a central web page for the service and entering the appropriate information to access the desired information.

4.9.8. Storage Mode

Photos are usually stored as files. They can then be organized into meta-structures (e.g. albums, galleries, etc.) using sophisticated web photo album generators.

4.10. P2P Cache

Caching of P2P traffic is a useful approach to reduce P2P network traffic, because objects in P2P systems are mostly immutable and the traffic is highly repetitive. In addition, making use of P2P caches do not require changes to P2P protocols and can be deployed transparently from clients.

P2P caches operate similarly to web caches, in that they temporarily store frequently-requested content. Requests for content already stored in the cache can be served from local storage instead of requiring the data to be transmitted over expensive network links.

Two types of P2P caches exist: non-transparent P2P caches and transparent P2P caches. A non-transparent cache appears as a super peer; it explicitly peers with other P2P clients. For a transparent cache, once a P2P cache is established, the network will transparently redirect P2P traffic to the cache, which either serves the file directly or passes the request on to a remote P2P user and simultaneously caches that data. Transparency is typically implemented using deep packet inspection (DPI). DPI products identify and pass P2P packets to the P2P caching system so it can cache the traffic and accelerate it.

To enable operation with existing P2P software, P2P caches directly support P2P application protocols. A large number of P2P protocols are used by P2P software, and hence are supported by caches, leading to higher complexity. Additionally, these protocols evolve over time, and new protocols are introduced.

4.10.1. Applicability to DECADE

Very widely used (deployed) example of in-network storage for P2P systems. The existence and operation of the storage is totally transparent to the end user.

4.10.2. Transparent P2P Caches

4.10.2.1. Data Access Interface

Data Access Interface allows P2P content to be cached (stored) and supplied (retrieved) locally such that network traffic is reduced, but it is transparent to P2P users, and P2P users implicitly use the data-access interface (in the form of their native P2P application protocol) to store or retrieve content.

4.10.2.2. Data Management Operations

Not provided.

4.10.2.3. Data Search Capability

Not provided.

4.10.2.4. Access Control Authorization

Access control method is typically public-restricted (to any client which is part of the P2P channel or swarm).

4.10.2.5. Resource Control Interface

Not provided.

4.10.2.6. Discovery Mechanism

Use of Deep Packet Inspection means no discovery mechanism provided to P2P users, it is transparent to P2P users. Since DPI is used to recognize P2P applications private protocols, P2P Cache is getting more and more complicated as the P2P applications keep evolving.

4.10.2.7. Storage Mode

Object-based. Chunks (typically, the unit of transfer amongst P2P clients) of content are stored in the cache.

4.10.3. Non-transparent P2P Caches

4.10.3.1. Data Access Interface

Data Access Interface allows P2P content to be cached (stored) and supplied (retrieved) locally such that network traffic is reduced. P2P users implicitly store and retrieve from the cache using the P2P application's native protocol.

4.10.3.2. Data Management Operations

Not provided.

4.10.3.3. Data Search Capability

Not provided.

4.10.3.4. Access Control Authorization

Access control method is typically public-restricted (to any client which is part of the P2P channel or swarm)

4.10.3.5. Resource Control Interface

Not provided.

4.10.3.6. Discovery Mechanism

Cache pretends to be normal peers to join the P2P overlay network. Other P2P users can find these cache nodes through overlay routing mechanism, just looking to them as normal neighbor nodes.

4.10.3.7. Storage Mode

Object-based. Chunks (typically, the unit of transfer amongst P2P clients) of content are stored in the cache.

4.11. Web Cache

Web cache [17] is a well-built technology since the late 1990s, which has been widely deployed by many ISPs to reduce bandwidth consumption and web access latency. A web cache can cache the web documents (e.g., HTML pages, images) between server and client to reduce bandwidth usage, server load, and perceived lag. A web cache server is typically shared by many clients, and stores copies of documents passing through it; subsequent requests may be satisfied from the cache if certain conditions are met.

Another form of cache is a client-side cache, typically implemented in web browsers. A client side cache can keep a local copy of all pages recently displayed by browser, and when the user returns to one of these web pages, the local cached copy is reused.

A related protocol for P2P applications to use web cache is HPTP (HTTP based Peer to Peer) [16]. It proposes to share chunks of P2P files/streams using HTTP protocol with cache-control headers.

4.11.1. Applicability to DECADE

Very widely used (deployed) example of in-network storage for the key Internet application of Web browsing. The existence and operation of the storage is transparent to the end user in most cases. The content caching time is controlled by Time To Live parameters associated with the original content. The principle of web caching is to speed up web page reading by using (the same) content previously requested by a preceding user to service a new user.

4.11.2. Data Access Interface

Users explicitly read from a web cache by making requests, but they cannot explicitly write data into it. Data is implicitly stored into the web cache by requesting content that not already cached and meets policy restrictions of the cache provider.

4.11.3. Data Management Operations

Not provided.

4.11.4. Data Search Capability

Not provided.

4.11.5. Access Control Authorization

Access control method is public-unrestricted for stored content. It is important to note that if content is authenticated or encrypted (e.g. HTTPS, SSL) it will not be cached. Also if the content is flagged as private (vs public) at the HTTP level by the origin server it will not be cached.

4.11.6. Resource Control Interface

Not provided.

4.11.7. Discovery Mechanism

Web Caches can be transparently deployed between Web Server and Web Clients, employing DPI for discovery. Alternatively, web caches could be explicitly discovered by clients using techniques such as DNS or manual configuration.

4.11.8. Storage Mode

Object based. Web content is keyed within the cache by HTTP Request fields, such as Method, URI, and Headers.

4.12. Usenet

Usenet is a distributed Internet based discussion (message) system. The Usenet messages are arranged as a set of "newsgroups" that are classified hierarchically by subject. Usenet information is distributed and stored among a large conglomeration of servers that store and forward messages to one another in so called news feeds. Individual users may read messages from and post messages to a local news server typically operated by an ISP. This local server communicates with other servers and exchanges articles with them. In this fashion, the message is copied from server to server and eventually reaches every server in the network. [26]

Traditional Usenet as described above operates as a P2P network between the servers, and in a client-server architecture between the user and their local news server. The user requires a Usenet client

to be installed on their computer and a Usenet server account (through their ISP). However, with the rise of web browsers the Usenet architecture is evolving to be web based. The most popular example of this is Google Groups where Google hosts all the newsgroups and client access is via a standard HTTP based web browser.[27]

4.12.1. Applicability to DECADE

A historically very important and widely used (deployed) example of in-network storage in the Internet. The use of this system is rapidly declining but efforts have been made to preserve the stored content for historical purposes.

4.12.2. Data Access Interface

Users can read and post (store) messages.

4.12.3. Data Management Operations

Users sometimes have limited ability to delete messages that they previously posted.

4.12.4. Data Search Capability

Traditionally, users could manually search through the newsgroups as they are classified hierarchically by subject. In the newer web based systems there is also automatic search capability based on key word matches.

4.12.5. Access Control Authorization

Access control method is either public-unrestricted or private (to members of that newsgroup).

4.12.6. Resource Control Interface

Not provided.

4.12.7. Discovery Mechanism

Usually by manually logging on to the Usenet account.

4.12.8. Storage Mode

Messages are usually stored as files which are then organized hierarchically by subject into newsgroups.

4.13. Observations Regarding In-Network Storage Systems

The majority of the surveyed systems were designed for client-server architectures and do not support P2P. However, there are some important exceptions, especially for some of the newer technologies such as BranchCache and P2P Cache which do support a P2P mode.

The P2P cache systems are interesting since they do not require changes to P2P applications themselves. However, this is also a limitation in that they are required to support each application protocol.

Many of the surveyed systems were designed for caching as opposed to long term network storage. Thus during DECADE protocol design it should be carefully considered if a caching mode should be supported in addition to a network storage mode. There is typically a trade-off between providing a caching mode and long-term (and usually also reliable) storage with regards to some performance metrics.

Today, most in-network storage systems follow some variant of the authorization model of public-unrestricted, public-restricted, and private. For DECADE, we may need to evolve the authorization model to support a resource owner (e.g. end user) authorization, in addition to the network authorization.

5. Storage And Other Related Protocols

This section surveys existing storage and other related protocols, as well as comments on the usage of these protocols to satisfy DECADE's use cases. The surveyed protocols are listed alphabetically.

5.1. iSCSI

Small Computer System Interface (SCSI) is a set of protocols enabling communication with storage devices such as disk drives and tapes; internet SCSI (iSCSI) [5] is a protocol enabling SCSI commands to be sent over TCP. As in SCSI, iSCSI allows an Initiator to send commands to a Target. These commands operate on the device level as opposed to individual data objects stored on the device.

5.1.1. Data Access Interface

Read and write commands indicate which data is to be read or written by specifying the offset (using Logical Block Addressing) into the storage device. The size of data to be read or written is an additional parameter in the command.

5.1.2. Data Management Operations

Since commands operate at the device level, management operations are different than with traditional file systems. Management commands for SCSI/iSCSI including explicit device control such as starting and stopping the device and formatting the device.

5.1.3. Data Search Capability

SCSI/iSCSI does not provide the ability to search for particular data within a device. Note that such capabilities can be implemented outside of iSCSI.

5.1.4. Access Control Authorization

With respect to access to devices, the access control method is private. iSCSI uses CHAP [3] to authenticate initiators and targets when accessing storage devices. However, since SCSI/iSCSI operates at the device level, neither authentication nor authorization are provided for individual data objects. Note that such capabilities can be implemented outside of iSCSI.

5.1.5. Resource Control Interface

Not provided.

5.1.6. Discovery Mechanism

Manual configuration may be used. An alternative is the internet Storage Name Service (iSNS) [6] provides the ability to discover available storage resources.

5.1.7. Storage Mode

Block-based. SCSI/iSCSI provides an Initiator block-level access to the storage device.

5.2. HTTP

HTTP [4] is a key protocol for the World Wide Web. It is a stateless client-server protocol that allows applications to be designed using the Representational State Transfer (REST) model. HTTP is often associated with downloading (reading) content from web servers to web browsers, but it also has support for uploading (writing) of content to web servers. It has been used as the underlying protocol for other protocols such as WebDAV.

HTTP is used in some of the most popular in-network storage systems

surveyed previously including CDNs, Photo Sharing, and Web Cache. Usage of HTTP by a storage protocol implies that no extra SW is required in the client (i.e. web based client) as all standard Web browsers are based on HTTP.

5.2.1. Data Access Interface

Basic read and write operations are supported (using HTTP GET, PUT and POST methods).

5.2.2. Data Management Operations

Not provided.

5.2.3. Data Search Capability

Not provided

5.2.4. Access Control Authorization

All methods of access control are supported: public-unrestricted, public-restricted and private.

The great majority of web pages are public-unrestricted in terms of reading but do not allow any uploading of content (i.e. are not in-storage systems). In-storage systems range from private or public-unrestricted for Photo Sharing described in Section 4.9.5 to public-unrestricted for Web Caching described in Section 4.11.5.

5.2.5. Resource Control Interface

Not provided.

5.2.6. Discovery Mechanism

Manual configuration is typically used. Clients address HTTP servers by providing a hostname.

5.2.7. Storage Mode

File-based storage (a non-collection resource can typically be thought of as a "file"). Files may be organized into collections, which typically map on to the HTTP Path hierarchy.

5.2.8. Comments

HTTP is based on a client-server architecture and thus is not directly applicable for the DECADE focus on P2P. Also, HTTP offers

only a rudimentary toolset for storage operations compared to some of the other storage protocols.

5.3. NFS

The Network File System is designed to allow users to access files over a network in a manner similar to how local storage is accessed. NFS is typically used in local area network or enterprise settings, though changes made in later versions of NFS make it easier to operate over the Internet.

5.3.1. Data Access Interface

Traditional file-system operations such as read, write, and update (overwrite) are provided. Locking is provided to support concurrent access by multiple clients.

5.3.2. Data Management Operations

Traditional file-system operations such as move and delete are provided.

5.3.3. Data Search Capability

User has the ability to list contents of directories to find filenames matching desired criteria.

5.3.4. Access Control Authorization

All methods of access control are supported: public-unrestricted, public-restricted and private. For example, files and directories can be protected using read, write, and execute permissions for the files owner, group, and the public (others). Also, NFSv4.1 has a rich ACL model allowing a list of Access Control Entries (ACEs) to be configured for each file or directory. The ACEs can specify per-user read/write access to file data, file/directory attributes, creation/deletion of files in a directory, etc.

5.3.5. Resource Control Interface

While disk space quotas can be configured, it typically limits the total amount of storage allocated to a particular user. User control of bandwidth and connections used by remote peers is not provided.

5.3.6. Discovery Mechanism

Manual configuration is typically used. Clients address NFS servers by providing a hostname and a directory that should be mounted.

5.3.7. Storage Mode

File-based storage, allowing files to be organized into directories.

5.3.8. Comments

The efficiency and scalability of the NFS access control method is a concern in the context of DECADE. In particular, Section 6.2.1 of [7] states that:

Only ACEs that have a "who" that matches the requester are considered.

Thus, in the context of DECADE, to specify per-peer access control policies for an object, a client would need to explicitly configure the ACL for the object for each individual peer. A concern with this approach is scalability when a client's peers may change frequently and ACLs for many small objects need to be updated constantly during participation in a swarm.

Note that NFS v4.1's usage of RPCSEC_GSS provides support for multiple security mechanisms. Kerberos V5 is required, but others such as X.509 Certificates are also supported by way of GSS-API. Note, however, that NFSv4.1's usage of such security mechanisms is limited to linking a requesting user to a particular account maintained by the NFS server.

5.4. OAuth

OAuth [14] is a protocol that richens the traditional client-server authentication model for web resources. In particular, OAuth distinguishes the "client" from the "resource owner", thus enabling a resource owner to authorize a particular client for access (e.g., for a particular lifetime) to private resources.

We include OAuth in this survey so that its authentication model can be evaluated in the context of DECADE. OAuth itself, however, is not a network storage protocol.

5.4.1. Data Access Interface

Not applicable.

5.4.2. Data Management Operations

Not applicable.

5.4.3. Data Search Capability

Not applicable.

5.4.4. Access Control Authorization

Not applicable. While similar in spirit to the WebDAV ticketing extensions [15], OAuth instead uses the following process: (1) a client constructs a delegation request, (2) the client forwards the request to the resource owner for authorization, (3) the resource owner authorizes the request, and finally (4) a callback is made to the client indicating that its request has been authorized.

Once the process is complete, the client has a set of token credentials that grant it access to the protected resource. The token credentials may have an expiration time, and they can also be revoked by the resource owner at any time.

5.4.5. Resource Control Interface

Not applicable.

5.4.6. Discovery Mechanism

Not applicable.

5.4.7. Storage Mode

Not applicable.

5.4.8. Comments

The ticketing mechanism requires server involvement and the discussion relating to WebDAV's proposed ticketing mechanism (see Section 5.5.8) applies here as well.

5.5. WebDAV

WebDAV [8] is a protocol designed for Web content authoring. It is developed as an extension to HTTP described in Section 5.2, meaning it can be simpler to integrate into existing software. WebDAV supports traditional operations for reading/writing from storage, as well as other constructs such as locking and collections which are important when multiple users collaborate to author or edit a set of documents.

5.5.1. Data Access Interface

Traditional read and write operations are supported (using HTTP GET and PUT methods, respectively). Locking is provided to ease concurrent access by multiple clients.

5.5.2. Data Management Operations

WebDAV supports traditional file-system operations such as move, delete and copy. Objects are organized into collections, and these operations can also be performed on collections. WebDAV also allows objects to have user-defined properties.

5.5.3. Data Search Capability

User has the ability to list contents of collections to find objects matching desired criteria. A SEARCH extension [9] has also been specified allowing listing of objects matching client-defined criteria.

5.5.4. Access Control Authorization

All methods of access control are supported: public-unrestricted, public-restricted and private.

For example, an ACL extension [11] is provided for WebDAV. ACLs allow both user- and group-based access control policies (relating to reading, writing, properties, locking, etc) to be defined for objects and collections.

A ticketing extension [15] has also been proposed, but has not progressed passed an Internet Draft. This extension allows a client to request the WebDAV server to create a "ticket" (e.g., for reading an object) that can be distributed to other clients. Tickets may be given expiration times, or may only allow for a fixed number of uses. The proposed extension requires the server to generate tickets and maintain state for outstanding tickets.

5.5.5. Resource Control Interface

An extension [10] allows disk space quotas to be configured for Collections. The extension also allows WebDAV clients to query current disk space usage. User control of bandwidth and connections used by remote peers is not provided.

5.5.6. Discovery Mechanism

Manual configuration is typically used. Clients address WebDAV servers by providing a hostname.

5.5.7. Storage Mode

File-based storage (a non-collection resource can typically be thought of as a "file"). Files may be organized into collections, which typically map on to the HTTP Path hierarchy.

5.5.8. Comments

The efficiency and scalability of the WebDAV access control method is a concern in the context of DECADE, for similar reasons as stated in Section 5.3.8 for NFS. The proposed WebDAV ticketing extension partially alleviates this concern, but the particular technique may need further evaluation before being applied to DECADE. In particular, since DECADE clients may continuously upload/download a large number of small-size objects, and a single DECADE server may need to scale to many concurrent DECADE clients, requiring the server to maintain ticket state and generate tickets may not be the best design choice. Server-generated tickets can also increase latency for data transport operations depending on the message flow used by DECADE.

5.6. Observations Regarding Storage and Related Protocols

All of the surveyed protocols were primarily designed for client-server architectures and not for P2P. However, it is conceivable that some of the protocols could be adapted to work in a P2P architecture.

Several popular in-network storage systems today use HTTP as their key protocol even though it is not classically considered as a storage protocol. HTTP is a stateless protocol that is used to design RESTful applications. HTTP is a well supported and widely implemented protocol which can provide important insights for DECADE.

The majority of the surveyed protocols do not support low latency access for applications such as live streaming. This was one of the key general requirements for DECADE.

The majority of the surveyed protocols do not support any form of resource control interface. Resource control is required for users to manage the resources on in-network storage that can be used by other peers, e.g., the bandwidth or connections. Resource control is a key capability required for DECADE.

Nearly all surveyed protocols did however support the following capabilities which are required for DECADE: user ability to read/write content; some form of access control; some form of error indication; and ability to traverse firewalls and NATs.

6. Conclusions

Though there have been many successful in-network storage systems, they have been designed for use cases different from those defined in DECADE. For example, many of the surveyed in-network storage systems and protocols were designed for client-server architectures and not P2P. No surveyed system or protocol has the functionality and features to fully meet the set of requirements defined for DECADE. DECADE aims to provide a standard protocol for P2P applications and content provider to access and control in-network storage, resulting in increased network efficiency while retaining control over content shared with peers. Additionally, defining a standard protocol can reduce complexity of in-network storage since multiple P2P application protocols no longer need to be implemented by in-network storage systems.

7. Security Considerations

This draft is a survey of existing in-network storage systems, and does not introduce any security considerations beyond those of the surveyed systems.

For more information on security considerations of DECADE, see [1].

8. IANA Considerations

This document does not have any IANA Considerations.

9. Acknowledgments

The authors would like to thank Haibin Song, Yu-Shun Wang and Ning Zong for comments and contributions to this document.

10. Revision History

The following changes were made in this draft when going from Rev 00 to Rev 01:

- 1) Changed the publication date to the current date.
- 2) Moved section 6 "Observations and Analysis" to section 4.12 and renamed it to "Observations Regarding In-Network Storage Systems". Also made various editorial changes to the wording in this section.
- 3) Added a new section 5.6 called "Observations Regarding Storage and Related Protocols" and moved some portions of the original section 6 here.
- 4) Made editorial changes of the wording of the Conclusions (section 8) to reflect the updated Observations sections.
- 5) Changed TOC depth to "2 levels" from "3 levels".
- 6) Added Google Storage as another example to section 4.1.
- 7) Updated wording of DTN Applicability text in 4.5.1
- 8) Added new section on Named Data Networking in 4.6
- 9) Various minor editorial changes (fixing typos and grammar, defining acronyms, etc.) throughout the document.

11. Informative References

- [1] Yongchao, S., Zong, N., Yang, Y., and R. Alimi, "DECoupled Application Data Enroute (DECADE) Problem Statement", draft-song-decade-problem-statement-02 (work in progress), July 2010.
- [2] Yingjie, G., Bryan, D., Yang, Y., and R. Alimi, "DECADE Requirements", draft-gu-decade-reqs-05 (work in progress), July 2010.
- [3] Simpson, W., "PPP Challenge Handshake Authentication Protocol (CHAP)", RFC 1994, August 1996.
- [4] 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.
- [5] Satran, J., Meth, K., Sapuntzakis, C., Chadalapaka, M., and E. Zeidner, "Internet Small Computer Systems Interface (iSCSI)", RFC 3720, April 2004.
- [6] Tseng, J., Gibbons, K., Travostino, F., Du Laney, C., and J.

- Souza, "Internet Storage Name Service (iSNS)", RFC 4171, September 2005.
- [7] Shepler, S., Eisler, M., and D. Noveck, "Network File System (NFS) Version 4 Minor Version 1 Protocol", RFC 5661, January 2010.
 - [8] Dusseault, L., "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)", RFC 4918, June 2007.
 - [9] Reschke, J., Reddy, S., Davis, J., and A. Babich, "Web Distributed Authoring and Versioning (WebDAV) SEARCH", RFC 5323, November 2008.
 - [10] Korver, B. and L. Dusseault, "Quota and Size Properties for Distributed Authoring and Versioning (DAV) Collections", RFC 4331, February 2006.
 - [11] Clemm, G., Reschke, J., Sedlar, E., and J. Whitehead, "Web Distributed Authoring and Versioning (WebDAV) Access Control Protocol", RFC 3744, May 2004.
 - [12] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, April 2007.
 - [13] Scott, K. and S. Burleigh, "Bundle Protocol Specification", RFC 5050, November 2007.
 - [14] Hammer-Lahav, E., "The OAuth 1.0 Protocol", RFC 5849, April 2010.
 - [15] Ito, K., "Ticket-Based Access Control Extension to WebDAV", draft-ito-dav-ticket-00 (work in progress), October 2001.
 - [16] G. Shen, Y. Wang, Y. Xiong, B.Y. Zhao, Z.-L. Zhang, "HPTP: Relieving the tension between isps and p2p", In 6th International workshop on Peer-To-Peer Systems (IPTPS2007).
 - [17] Geoff Huston, Telstra., "Web Caching", In The Internet Protocol Journal Volume 2, No. 3.
 - [18] Pathan, A.K., Buyya, R., "A Taxonomy and Survey of Content Delivery Networks", In Grid Computing and Distributed Systems Laboratory in University of Melbourne, Technology Report, Feb. 2007.
 - [19] Amazon, "Amazon Simple Storage Service (Amazon S3)",

<http://aws.amazon.com/s3/>.

- [20] Microsoft Corporation., "Windows Azure Blob - Programming Blob Storage".
- [21] Google, "Google Storage for Developers", <http://code.google.com/apis/storage>.
- [22] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiawicz., "Pond: the OceanStore Prototype", In FAST 2003.
- [23] A. Anand, V. Sekar, A. Akella., "SmartRE: An Architecture for Coordinated Network-wide Redundancy Elimination", In SIGCOMM 2009.
- [24] S. Paul, R. Yates, D. Raychaudhuri, J. Kurose., "The Cache-and-Forward Network Architecture for Efficient Mobile Content Delivery Services in the Future Internet", In Innovations in NGN: Future Network and Services, 2008.
- [25] Microsoft Corporation., "BranchCache", <http://technet.microsoft.com/en-us/network/dd425028.aspx>.
- [26] Wikipedia, "Usenet", <http://en.wikipedia.org/wiki/Usenet>.
- [27] Google, "Google Groups", <http://groups.google.com>.
- [28] Kodak, "Kodak Gallery Home Page", <http://www.kodakgallery.com/gallery/welcome.jsp>.
- [29] Wikipedia, "Kodak Gallery", http://en.wikipedia.org/wiki/Kodak_Gallery.
- [30] Flickr, "Flickr Home Page", <http://www.flickr.com>.
- [31] ImageShack, "ImageShack Home Page", <http://imageshack.us>.
- [32] Tumblr, "Tumblr Home Page", <http://www.tumblr.com>.
- [33] Named Data Networking, "Named Data Networking Home Page", <http://www.named-data.net/>.
- [34] Named Data Networking, "Named Data Networking Project Proposal", <http://www.named-data.net/ndn-proj.pdf>.

Appendix A. Authors

[[Comment.1: RFC Editor: Please move information in this section to the Authors' Addresses section at publication time.]]

ZhiHui Lu
Fudan University

Email: lzh@fudan.edu.cn

Pang Tao
China Telecom

Email: pangt@gsta.com

Juan Carlos Zuniga
InterDigital Communications, LLC

Email: JuanCarlos.Zuniga@InterDigital.com

Authors' Addresses

Richard Alimi (editor)
Yale University

Email: richard.alimi@yale.edu

Akbar Rahman (editor)
InterDigital Communications, LLC

Email: Akbar.Rahman@InterDigital.com

Yang Richard Yang (editor)
Yale University

Email: yry@cs.yale.edu

Individual Submission
Internet-Draft
Intended status: Informational
Expires: April 28, 2011

B. Ohlman
Ericsson
O. Strandberg
Nokia Siemens Networks
C. Dannewitz
University of Paderborn
A. Lindgren
SICS
R. Maglione
Telecom Italia
B. Ahlgren
SICS
D. Kutscher
NEC
October 25, 2010

Requirements for accessing data in network storage
draft-ohlman-decade-add-use-cases-reqs-02

Abstract

The DECoupled Application Data Enroute (DECADE) working group is specifying standardized interfaces for accessing in-network storage from applications to store, retrieve and manage data. The main objective is to provide a framework that is useful to P2P applications, without excluding other, possibly related applications that can benefit from accessing in-network storage. This memo presents Internet TV as a specific application scenario where access to in-network storage would be required and lists a set of concrete requirements that should be considered for the DECADE architecture and protocol specifications.

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 [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 April 28, 2011.

Copyright Notice

Copyright (c) 2010 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
2. Internet TV Scenario	5
2.1. Detailed Scenario Description	5
2.2. Summary	6
3. Specific requirements	6
3.1. Unique Naming of Information Objects	6
3.1.1. Requirement	6
3.1.2. Rationale	7
3.1.3. Discussion	7
3.2. Access to Information Objects	7
3.2.1. Requirement	7
3.2.2. Rationale	7
3.2.3. Discussion	7
3.3. Real-time Support	7
3.3.1. Requirement	7
3.3.2. Rationale	7
3.3.3. Discussion	8
3.4. Discovery service for DECADE in-network storage	8
3.4.1. Requirement	8
3.4.2. Rationale	8
3.4.3. Discussion	8
3.5. Multiple active DECADE Storage Servers	8
3.5.1. Requirement	8
3.5.2. Rationale	8
3.5.3. Discussion	8
4. IANA Considerations	9
5. Security Considerations	9
6. Acknowledgments	9
7. Informative References	9
Authors' Addresses	9

1. Introduction

The DECADE approach to access to in-network storage through standardized interfaces has been motivated by P2P application scenarios where it can be beneficial to refer peers to in-network storage system in a convenient network-topological location in order to enhance data exchange for a given P2P application.

One specific example would be a P2P application instance in a home network connected via an ADSL access network. In a traditional P2P approach, this application instance would download chunks from other peers and serve chunks to other peers in parallel. With an asymmetric uplink, the application instance (and other hosts on the home network) are likely to experience uplink congestion, if the application instance is selected by a substantial number of peers and has to serve data to them. In situations with prevalent asymmetric access links, the P2P session (all peer application instances) would also be limited in the achievable downlink speed because the aggregate uplink bandwidth is not sufficient.

DECADE in-network storage servers are expected to be helpful in such scenarios, because peers could be referred to appropriate storage servers for downloading some content, thereby offloading traffic from the capacity-limited access uplinks.

Such storage servers are expected to provide interfaces for storing, retrieving, and managing data. The general concept is that, in a distributed P2P application, some instance would upload data to a storage server and then refer other instances, such as P2P peers, to that data, for instance by passing a certain URI. In addition, there could be interactions for allocating capacity on servers for certain applications, for deleting data etc.

This document argues that, while such a system would be very useful for P2P applications, there are others, related applications that could benefit from in-network storage in just the same way. Though only applications that does not lead to a completely new set of requirements should be taken into account. This document argues that it should be possible to extend the DECADE work to include certain other applications without increasing the overall complexity of the solution.

Specifically, this document describes the in-network-storage aspects of Internet TV -- an important application today, that can significantly benefit from in-network-caching. We argue that it seems negligent to exclude Internet TV from leveraging DECADE in-network-storage and describe specific use cases for accessing in-network-storage in such a scenario. Moreover, we present a set of

requirements that should be met by the DECADE architecture design in order guarantee its applicability to such applications. We propose that these requirements be added to the DECADE requirements specification.

In this memo, we align the requirement description to the layout used in [I-D.gu-decade-reqs]. Section 2 describes the Internet TV scenario, and Section 3 presents corresponding requirements that should be considered to ensure a broad enough applicability of the DECADE framework.

2. Internet TV Scenario

Internet TV is a general term to refer to different kinds of systems or applications where video is delivered to (mostly) home network devices for immediate rendering or storing. In this memo, we refer to the distribution of video content, mainly focusing on Video-on-Demand (VoD) services and user-generated content.

VoD services are commonly widespread in many service providers' networks. This scenario is characterized by the need to support an efficient large-scale distribution of video, possibly with a fairly high degree of replicated contents, to a multiplicity of fixed and mobile users. By supporting this application with DECADE protocols, video content can be retrieved from the in-network storage, achieving a number of benefits. The originating servers can be relieved from most of the load, since popular content will be automatically available in the in-network storage, closer to the users. Improved network efficiency will be achieved, reducing the traffic load in the upstream network segments. Moreover user experience, also for mobile users, can be improved.

2.1. Detailed Scenario Description

A well-known issue with Internet TV applications such as YouTube is the flash crowd problem. That is also an example of a problem which could be significantly eased if in-network storage is used to provide users with locally available copies rather than all requesting the data from the source. This can be extra beneficial for services with real-time (or near real-time) components as traditional pre-caching solutions can be difficult to use then.

A particular interesting Internet TV variant is "hybrid Internet TV" based on an Internet TV distribution service that is a hybrid between traditional CDN and a P2P service. Such a service would distribute content from central servers, make use of CDN caches on the way and finally use the end hosts/STB as caches for the P2P part of the

application.

If only the P2P application in the host/STB can store content in the DECADE storage, the content first has to be downloaded from the Internet TV server/CDN cache over the access link to the host/STB and then uploaded, over the same access link, to the DECADE storage before any peer in the P2P part of the application can access it from the DECADE storage (instead of downloading it from the client).

To avoid this, it should be possible for the DECADE storage to 'cache' the content when the first download to the host/STB is done. That would mean the content never have to travel over the capacity-limited uplink. For this to be feasible, one requirement is that the Internet TV service can prompt the DECADE storage that certain content should be cached on its way to the host. Having such functionality, that allows a host to get content from the DECADE storage of neighboring host rather than from a central server, would of course also offload the core network in the same way a traditional CDN does.

2.2. Summary

The DECADE architecture and protocol specifications should take the hybrid Internet TV scenario into account to ensure a reasonable level of generality of the DECADE in-network storage. While P2P-specific requirements should be considered, DECADE should not be unnecessarily limited to it.

Specifically, dissemination applications of streaming type (some with real-time or close to real-time requirements) should be supported by DECADE as they can cause significant load on the network. The network load could be reduced significantly for these types of applications if copies stored locally in the network could be used instead of always fetching data from the source.

3. Specific requirements

3.1. Unique Naming of Information Objects

3.1.1. Requirement

When a DECADE client in a certain application context stores an information object in DECADE storage servers, the object **MUST** be addressable by a unique name across different application contexts.

3.1.2. Rationale

There is a need for unique naming to enable different application instances to refer to information objects using a name (that may have been provided to them by another DECADE client). Such unique naming is essential for efficient cache handling and can serve for de-duplication.

3.1.3. Discussion

Unique naming can be achieved in different ways. Names can be assigned from some (structured) names, for instance by URIs. Names can also be generated, for instance by calculating hashes of the object's content.

The detailed syntax and semantics of DECADE names (and the actual standardization requirements) are for further study.

3.2. Access to Information Objects

3.2.1. Requirement

It **MUST** be possible to access data stored on DECADE storage servers as complete information objects, such as a named video file.

3.2.2. Rationale

In a video-on-demand caching use case, the client application should be enabled to retrieve the complete object in one transaction and should not be required to download individual chunks.

3.2.3. Discussion

This does not necessarily impose implications on the way that the storage servers store the object.

3.3. Real-time Support

3.3.1. Requirement

The DECADE storage service **MUST** support real-time applications in a way that a resource that is being uploaded is already available for download.

3.3.2. Rationale

For larger objects or chunks, it is not acceptable if a DECADE client has to upload the complete resource first, before other clients can

start downloading it.

3.3.3. Discussion

This requirement should also be important for P2P live streaming.

3.4. Discovery service for DECADE in-network storage

3.4.1. Requirement

When a DECADE client attach to a DECADE enabled network there SHOULD be a discovery service that can tell a DECADE client where in-network storage servers can be found.

3.4.2. Rationale

To minimize manual configuration of the DECADE clients, a discovery service, similar to DHCP , should be provided in the DECADE enabled network.

3.4.3. Discussion

In particular, this simplifies the administration of the DECADE in-network storage for a user that roams to a visited network.

3.5. Multiple active DECADE Storage Servers

3.5.1. Requirement

A DECADE client SHOULD be able to use multiple in-network storage servers at the same time.

3.5.2. Rationale

One example of when this is needed is when a user/client roams to another network, then it is reasonable to assume that the currently used in-network storage remains active for a certain time not to disrupt ongoing communication sessions at the same time as another in-network storage might immediately be needed in the new network.

3.5.3. Discussion

A user of DECADE in-network storage who roams to a visited network could potentially cause very inefficient access to that user's DECADE storage. It is therefore essential that the user is able to acquire new DECADE storage which is better located in the visited network. Usage that could result in such inefficiencies is communication with other users locally in the same network, for example as part of a

small meeting or large event (fair, sports event, etc).

A related issue is the possibility to migrate content from one DECADE storage to another when roaming. We believe that this is covered by the requirements on Efficient Transfer (section 3.3) and Communication among In-network Storage Elements (section 4.3) of [I-D.gu-decade-reqs].

4. IANA Considerations

This document has no requests to IANA.

5. Security Considerations

The re-use of copies in the network part of DECADE will require that appropriate access control mechanisms are designed.

6. Acknowledgments

We would like to thank all persons participating in the Network of Information work package in the EU FP7 projects 4WARD and SAIL for contributions and feedback to this document.

7. Informative References

[I-D.gu-decade-reqs]

Yingjie, G., Bryan, D., Yang, Y., and R. Alimi, "DECADE Requirements", draft-gu-decade-reqs-05 (work in progress), July 2010.

[RFC2119]

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

Authors' Addresses

Borje Ohlman
Ericsson
Stockholm
Sweden

Email: Borje.Ohlman@ericsson.com

Ove Strandberg
Nokia Siemens Networks
Linnoitustie 6
Espoo
Finland

Email: ove.strandberg@nsn.com

Christian Dannewitz
University of Paderborn
Paderborn
Germany

Email: cdannewitz@upb.de

Anders Lindgren
Swedish Institute of Computer Science
Stockholm
Sweden

Email: andersl@sics.se

Roberta Maglione
Telecom Italia
Turin
Italy

Email: roberta.maglione@telecomitalia.it

Bengt Ahlgren
Swedish Institute of Computer Science
Stockholm
Sweden

Email: bengta@sics.se

Dirk Kutscher
NEC Laboratories Europe
Kurfuersten-Anlage 36
Heidelberg
Germany

Email: kutscher@neclab.eu

DECADE
Internet Draft
Intended status: Informational
Expires: April 2011

Xin Wang
Fudan University
Jin Zhao
Fudan University
Tiegang Zeng
Fudan University
Jun Li
Fudan University
Lei Liu
Fudan University
Shihui Duan
China CATR
October 25, 2010

Router-supported Data Regeneration for In-network Storage Systems
draft-wang-decade-data-regeneration-01.txt

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), 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 April 25, 2011.

Copyright Notice

Copyright (c) 2010 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.

Abstract

In-network storage systems store redundancy to compensate for the data loss incurred by hardware failures or other reasons. This document introduces a practical work of router-supported data regeneration in in-network storage systems to maintain the amount of redundancy. This proposed regeneration process can exploit the bandwidth diversity in the network, and the corresponding protocol enables supporting routers work transparently to support the regeneration process.

Table of Contents

1. Introduction	4
2. Key problems	4
3. Overview of the Router-Supported Regeneration Process	6
3.1. Regeneration Process	6
3.2. File Regeneration Protocol	7
3.3. System Implementation and Components	9
3.4. Key Technologies	10
4. Validation	11
5. DECADE Compatibility.....	12
6. Security Considerations	12
7. IANA Considerations	12
8. Conclusions	13
9. References	13
9.1. Normative References	13
9.2. Informative References	13

1. Introduction

In-network storage systems store a substantial volume of data in an overlay network containing a large number of storage servers, which can be used for online storage service, such as file sharing, CDN, and etc. In such systems, peer churns, hardware failures and other malfunctions are unavoidable so that some data may not be accessible. Thus, the system should maintain a certain ratio of redundancy so that a subset of data is enough for data recovery.

Storing coded data of original files rather than their replicas [1] can maintain higher data integrity [2], so that any k of n storage servers can retrieve the original data. On the other hand, if a storage server fails, a replacement server should regenerate the data stored in the failed server. For coded data, this way guarantees that any k servers can retrieve the original file. The replacement server, which we call as "newcomer" in this document, should contact at least k storage servers, which we call as "providers" in this document.

The efficiency of such regeneration is influenced by the topology of the network. First, since the newcomer should contact multiple providers, several flows of data from providers may converge at some links, and thus incur a significant bottleneck in the transmission. Second, the topology may have an influence on the available bandwidth between two servers. For example, two servers in a subnet may probably have higher available bandwidth between them than two servers in two different subnets. As shown in [3], the diversity of bandwidth incurred by network topology can be exploited by letting providers relay the data flow from other providers to form a tree-structured topology during the regeneration, where network coding naturally resides, such that some slow links can be bypassed.

In this document, we show that the devices which relay the traffic during the regeneration can be not only providers, but routers as well. Routers can support the regeneration, as it can encode data when multiple data flows converge and reduce the overall traffic during the regeneration. Since the redundant maintenance is independent of data access, the scheme that we present is compatible with the protocol DECADE to access in-network storage [4]. We show that routers can support the regeneration process transparently such that no storage servers should be aware of such routers or the network topology.

2. Key problems

In order to support data regeneration in in-network storage systems, routers should be able to work transparently so that storage servers

participating in the regeneration do not need any information about routers in the network. To satisfy this goal, the following key problems should be considered:

- a) Bandwidth: During the regeneration, since providers are allowed to relay the traffic from other providers, available bandwidth between servers should be measured to determine the optimal routing. However, since supporting routers are supposed to be transparent to storage servers, we can only measure the end-to-end available bandwidth between each two servers participating in the regeneration process.
- b) Routing: Since we can only get the table of the available bandwidth between each pair of participating servers, we first determine the routing on the overlay network covering the participating servers, then the transmission rate during the regeneration is optimized.
- c) Mapping: To make the supporting routers work, we need a mapping mechanism to make supporting routes aware of the regeneration process and know how to act during the regeneration. After the routing in the overlay network has been determined, participating servers may send data to their next-hop servers, to make supporting routers between them know that there will be traffic of a regeneration process coming soon. Supporting routers should be able to determine how many flows will come, whether it is necessary to encode such flows and where the next-hop is. After mapping, the data transmission can start.
- d) Reliability: Data transmission is unreliable in the network due to packet loss, disorder or other failures. Supporting routers should have a mechanism to make sure the data transmission is reliable. If the transmission between two servers is based on TCP, supporting routers should maintain the TCP state of incoming flows. If the transmission is based on UDP, there should be application-level retransmission scheme to guarantee the data reliability.
- e) Congestion control: TCP provides a mechanism to control the congestion in the network. However, if multiple flows are encoded by a supporting router, the router should control the congestion in place of the destination server. However, if the transmission is based on UDP, participating servers should perform end-to-end congestion control.

3. Overview of the Router-Supported Regeneration Process

Apart from data access, data regeneration is a part of functions provided by in-network storage systems. Our scheme requires that coded data are stored in the systems, and a file is divided into k blocks and n encoded blocks are produced after encoding in which any k blocks can recover the original file. The coding technique should be decentralized and the coding operations are not necessary to perform on a single server, such as random linear coding [5]. The n encoded blocks are stored in n storage servers, i.e., each storage server stores one encoded block. When a server fails, a replacement server, called newcomer, should contact at least k encoded blocks, and reconstruct a new encoded block by re-encoding these encoded blocks.

3.1. Regeneration Process

Data regeneration process should be carried out as follows.

1. A server failure is detected and then a regeneration process is triggered. One newcomer and at least k providers are selected. A weighted complete graph covering all these $k + 1$ servers is made in which the weight of each edge denotes the available bandwidth measured between each pair of the $k + 1$ servers. A maximum spanning tree, called regeneration tree, is constructed on such a complete graph.
2. The newcomer obtains IP addresses of all providers and the regeneration tree. It then sends a NOTIFICATION messages to each provider.
3. Each provider replies an ACK message when it receives a NOTIFICATION message to its parent in the regeneration tree.
4. When an ACK message goes through a supporting router, the supporting router forwards this message and stores IP addresses of the source and the destination of the ACK message. An operation table should be constructed on the supporting router that contains the sources and destinations of the received ACK message and the number of hops to the corresponding destinations.
5. Non-leaf providers modify the type of the received ACK message to a DACK message and forward it to the newcomer. If the newcomer has received ACK or DACK message from all providers, it sends a DETECT message to each provider.

6. When a provider receives a DETECT message, it replies a RE-DETECT message to its parent in the regeneration tree. When a RE-DETECT message goes through a supporting router, the supporting router select the destination with the minimum number of hops in the operation table as the new destination of the RE-DETECT message and then forwards it. The supporting router selects IP addresses of sources of all received RE-DETECT messages and construct an encoding table.
7. Non-leaf providers modify the type of the received RE-DETECT message to a DRE-DETECT message and forward it to the newcomer. If the newcomer has received RE-DETECT or DRE-DETECT messages from all providers, it sends a START message to each provider.
8. All providers begin to send data in DATA messages that contain fixed number of bits of data to its parent node when it has received a START message. A provider that has incoming flow(s) has to wait to send its first DATA message until the first DATA message of each incoming flow has arrived and it has encoded the received data with the data it stores.
9. When a DATA message goes through a supporting router, the supporting router stores it until it has received corresponding DATA messages from all entries in its encoding table. It then encodes the data in the received DATA message and sends a new DATA message that contains the encoded data to the destination with the minimum number of hops in the operation table.
10. The newcomer stores the received data. If the newcomer has multiple incoming flows, it encodes the received data and stores them. The regeneration finishes when it has received all data.

3.2. File Regeneration Protocol

The File Regeneration Protocol (FRP) runs on the application layer, as shown in Figure 1.

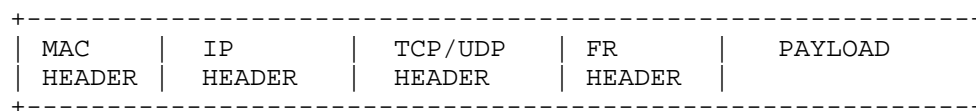


Figure 1 The overall structure of the File Regeneration Protocol

The structure of the FR head is shown as Figure 2.

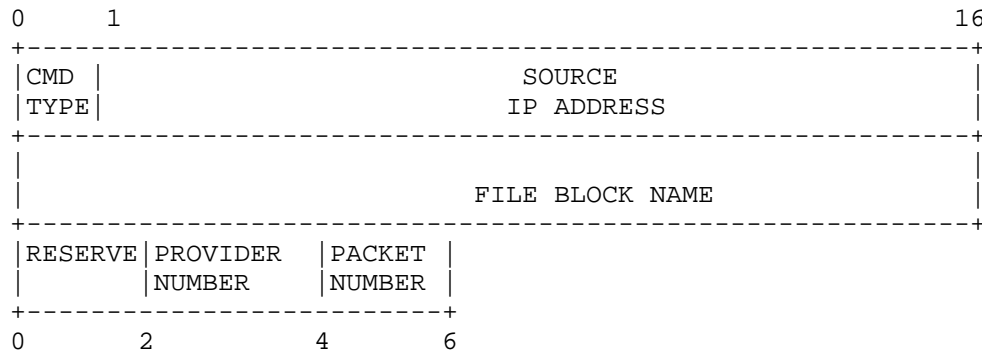


Figure 2 The structure of the FR header

"CMD TYPE" indicates the type of the message that includes:

NOTIFICATION: The newcomer sends a NOTIFICATION messages to providers to start a regeneration process.

ACK: A provider replies an ACK message to the newcomer when it receives a NOTIFICATION message. An ACK message makes supporting routers that it goes through be aware of the regeneration process.

DACK: A DACK message is no different from a ACK message except for the CMD TYPE. Supporting routers will not process the DACK message.

DETECT: The newcomer sends DETECT messages to providers when it has received ACK messages that contain addresses of all providers.

RE-DETECT: A provider replies an DETECT message to the newcomer when it receives a DETECT message. A RE-DETECT message makes supporting routers be aware of the number of incoming flows during the upcoming data transmission.

DRE-DETECT: A DRE-DETECT message is no different from a RE-DETECT message except for the CMD TYPE. Supporting routers will not process the DRE-DETECT message.

START: The newcomer sends START messages to all providers, indicating all servers are ready.

DATA: Providers send DATA messages that contain fixed number of bits of data to its parent in the regeneration tree. DATA messages may be encoded at supporting routers and are finally forwarded to the newcomer.

"SOURCE IP ADDRESS" represents the IP address of the last encoding device, which may be a provider or a supporting router.

"FILE BLOCK NAME" represents the name of the file block in the transmission.

"RESERVE" represents the segment that is reserved for future applications.

"PROVIDER NUMBER" represents the identifier number of the provider.

"PACKET NUMBER" represents the sequential number of the file block in the transmission.

3.3. System Implementation and Components

Router-supported data regeneration is an independent component in in-network storage systems. Since there are a large number of storage servers in the system, the server failure occurs frequently. To maintain the data integrity, a high-efficient mechanism is necessary to regenerate the lost data when a server fails. The implementation of our proposed in-network storage system contains two parts: storage servers and supporting routers. From the perspective of functions, storage servers are composed of three functional parts: dispatcher, newcomer and provider. Figure 3 illustrates the system architecture and components of the router-supported data regeneration.

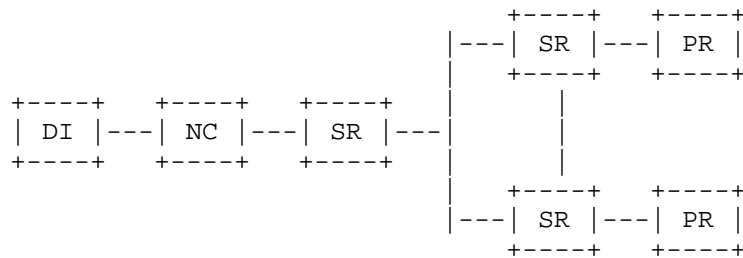


Figure 3 The architecture and components of the router-supported data regeneration

DISPATCHER (DI): It manages the whole system, including the selection of the newcomer and providers and the detection of server failures.

NEWCOMER (NC): It starts the regeneration process with providers and accepts data from providers. It encodes the received data and stores the encoded data as a regenerated block.

PROVIDER (PR): Providers are storage server that provider data to the newcomer in the regeneration process.

SUPPORTING ROUTER (SR): Supporting routers are routers with computing and cache capabilities and can support regeneration. Before the data transmission during the regeneration, it collects information from ACK and RE-DETECT message to be aware of the incoming flows and the destination in the regeneration process.

3.4. Key Technologies

Some key technologies are presented in this section, which can make data transmission rate improved, the bandwidth consumption saved and the spent time reduced during the regeneration.

1. When the newcomer and providers have been selected, we measure the available bandwidth between each pair of the newcomer and providers. A complete graph covering the newcomer and providers can be constructed and the weight of each edge is the corresponding available bandwidth between two servers. A maximum spanning tree, i.e., a regeneration tree, then is constructed on this graph, in which the newcomer is the root. All non-root servers in the regeneration tree send their data to its parent. When a flow of data transferred goes through a supporting router, it may be encoded with other flows and forwarded to another server. Compared with conventional regeneration process, the method we propose utilizes the link with higher available bandwidth in the network, reduces the communication cost and thus increases the transmission rate during the regeneration.
2. Another key technology in the router-supported data regeneration is data encoding on the supporting router. During the regeneration, supporting router detects File Regeneration Protocol (FRP) by processing all IP packets that go through it. Supporting routers recognizes the file block being regenerated by analyzing the FRP. If multiple flows of the same block come during the regeneration, a supporting router should encode the received data. The header of FRP enables the supporting router to know whether encoding operations should be performed. Utilizing the computing capability, encoding operations that should have been performed on the newcomer or providers are partially transferred to supporting routers, such that supporting routers sends out only one data flow even if it receives multiple data flows. Therefore, multiple data flows sharing the same physical link are eliminated or at least partially eliminated, and transmission rate during the regeneration can be significantly improved.

4. Validation

We present our evaluation results of our proposed scheme here. We implement supporting routers on servers running Linux (kernel 2.6.30). In the network, there are four supporting routers and four storage servers, among which one is selected as both the dispatcher and the newcomer and others are providers. The storage servers and supporting routers can connect in a topology shown in Figure 4.

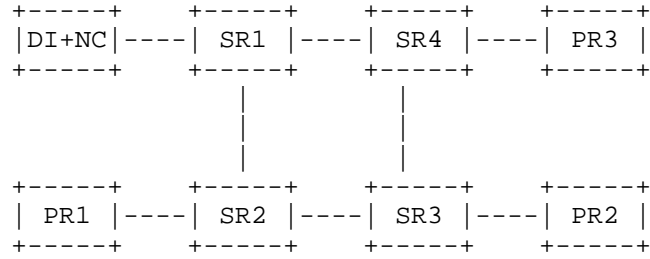


Figure 4 the network topology in the experiment

Each link in the network topology refers to a fast Ethernet (100BASE-TX, specifically). Actually we control the available bandwidth on each link as follows.

Table 1 The available bandwidth in the network topology

Node 1	Node 2	Available Bandwidth (Mbps/S)
DI+NC	SR1	60
SR1	SR4	50
SR4	PR3	80
SR1	SR2	40
SR1	SR3	25
SR4	SR3	20
PR1	SR2	80
SR2	SR3	50
SR3	PR2	80

We regenerate a coded block with a size of 6,000,000 bytes. We compare the time spent in the regeneration process and the bandwidth consumed between the conventional regeneration process in which the newcomer receives data directly from providers and the regeneration process we propose above. The experiment is repeated for 100 times and the result is the average value.

Table 2 The average bandwidth consumption

conventional regeneration	router-supported regeneration
58241047 byte	45606648 bytes

Table 2 shows the bandwidth consumption on average. We count the total number of bytes all providers and supporting routers send out. We can see that router-supported regeneration process is able to reduce the bandwidth consumption by 21.7%, since supporting routers can encode data from multiple devices.

Table 3 The average regeneration time

conventional regeneration	router-supported regeneration
95.4 sec.	49.4 sec.

Table 3 shows the average regeneration time. Router-supported regeneration can reduce the regeneration time by 48.3%, because it can not only reduce the bandwidth consumption, but also utilize the network topology by bypassing links with low available bandwidth.

5. DECADE Compatibility

Since in the in-network storage system, servers are not guaranteed to be stable, it is necessary to maintain the data integrity by regenerating the lost block after server failures. Thus, the File Regeneration Protocol (FRP) can work as a part of DECADE protocol. According to the reliability level of the applications, DECADE-compatible applications can implement FRP independently, which will not interfere with other part of the DECADE protocol.

6. Security Considerations

This draft does not introduce any new security issues.

7. IANA Considerations

This memo includes no request to IANA.

8. Conclusions

We propose a topology-aware regeneration process for in-network storage system such that bandwidth diversity in the network can be exploited and routers may support the regeneration process by encoding the incoming data flows. We present the corresponding protocol to configure the regeneration process adaptively in which supporting routers and servers do not need to know the network topology and make decisions by their local information. System architecture is presented and related key technologies are discussed.

9. References

9.1. Normative References

- [1] S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, C. Beame, M. Eisler, and D. Noveck, "Network File System (NFS) version 4 Protocol", RFC 3510, 2003.

9.2. Informative References

- [2] H. Song, N. Zong, Y. Yang, and R. Alimi, "DECoupled Application Data Enroute (DECADE) Problem Statement," <http://tools.ietf.org/id/draft-ietf-decade-problem-statement-00.txt>
- [3] H. Weatherspoon and J. Kubiatowicz, "Erasure Coding vs. Replication: A Quantitative Comparison," *Peer-to-Peer Systems*, vol. 2429/2002, pp. 328-337, 2002.
- [4] J. Li, S. Yang, X. Wang, and B. Li, "Tree-structured Data Regeneration in Distributed Storage Systems with Regenerating Codes," in *Proc. IEEE INFOCOM*, 2010.
- [5] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The Benefits of Coding over Routing in a Randomized Setting," in *Proc. International Symp. Inform. Theory*, pp. 442, 2003.

Authors' Addresses

Xin Wang
Fudan University
Shanghai 201203, China
Phone: 86-21-51355526
Email: xinw@fudan.edu.cn

Jin Zhao
Fudan University
Shanghai 201203, China
Phone: 86-21-51355526
Email: jzhao@fudan.edu.cn

Tiegang Zeng
Fudan University
Shanghai 201203, China
Phone: 86-21-51355526
Email: 09210240087@fudan.edu.cn

Jun Li
Fudan University
Shanghai 201203, China
Phone: 86-21-51355526
Email: 0572222@fudan.edu.cn

Lei Liu
Fudan University
Shanghai 201203, China
Phone: 86-21-51355526
Email: 09210240117@fudan.edu.cn

Shihui Duan
CATR
Beijing 100045, China
Phone: 86-10-63200068
Email: duanshihui@catr.cn

