Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 19, 2017

P. Hoffman
ICANN
P. McManus
Mozilla

June 17, 2017

DNS Queries over HTTPS
draft-hoffman-dispatch-dns-over-https-00

Abstract

DNS queries sometimes experience problems with end to end connectivity at times and places where HTTPS flows freely.

HTTPS provides the most practical mechanism for reliable end to end communication. Its use of TLS provides integrity and confidentiality guarantees and its use of HTTP allows it to interoperate with proxies, firewalls, and authentication systems where required for transit.

This document describes how to run DNS service over HTTP using https:// URIs.

[This paragraph is to be removed when this document is published as an RFC] Comments on this draft can be sent to the DNS over HTTP mailing list at https://www.ietf.org/mailman/listinfo/dnsoverhttp .

Status of This Memo

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

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

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

This Internet-Draft will expire on December 19, 2017.

Table of Contents

1. Introduction

The Internet does not always provide end to end reachability for native DNS. On-path network devices may spoof DNS responses, block DNS requests, or just redirect DNS queries to different DNS servers that give less-than-honest answers.

Over time, there have been many proposals for using HTTP and HTTPS as a substrate for DNS queries and responses. To date, none of those

proposals have made it beyond early discussion, partially due to disagreement about what the appropriate formatting should be and partially because they did not follow HTTP best practices.

This document defines a specific protocol for sending DNS [RFC1035] queries and getting DNS responses over modern versions of HTTP [RFC7540] using https:// (and therefore TLS [RFC5246] security for integrity and confidentiality).

The described approach is more than a tunnel over HTTP. It establishes default media formatting types for requests and responses but uses normal HTTP content negotiation mechanisms for selecting alternatives that endpoints may prefer in anticipation of serving new use cases. In addition to this media type negotiation, it aligns itself with HTTP features such as caching, proxying, and compression.

The integration with HTTP provides a transport suitable for both traditional DNS clients and native web applications seeking access to the DNS.

2. Terminology

A server that supports this protocol is called a "DNS API server" to differentiate it from a "DNS server" (one that uses the regular DNS protocol). Similarly, a client that supports this protocol is called a "DNS API client".

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

3. Use Cases

There are two primary use cases for this protocol.

The primary one is to prevent on-path network devices from interfering with native DNS operations. This interference includes, but is not limited to, spoofing DNS responses, blocking DNS requests, and tracking. HTTP authentication and proxy friendliness are expected to make this protocol function in some environments where DNS directly on TLS ([RFC7858]) would not.

A secondary use case is web applications that want to access DNS information. Standardizing an HTTPS mechanism allows this to be done in a way consistent with the cross-origin resource sharing [CORS] security model of the web and also integrate the caching mechanisms

of DNS with those of HTTP. These applications may be interested in using a different media type than traditional clients.

[This paragraph is to be removed when this document is published as an RFC] Note that these use cases are different than those in a similar protocol described at [I-D.ietf-dnsop-dns-wireformat-http]. The use case for that protocol is proxying DNS queries over HTTP instead of over DNS itself. The use cases in this document all involve query origination instead of proxying.

4. Protocol Requirements

The protocol described here bases its design on the following protocol requirements:

- The protocol must use normal HTTP semantics.
- The queries and responses must be able to be flexible enough to express every normal DNS query.
- The protocol must allow implementations to use HTTP's content negotiation mechanism.
- The protocol must ensure interoperable media formats through a mandatory to implement format wherein a query must be able to contain one or more EDNS extensions, including those not yet defined.
- The protocol must use a secure transport that meets the requirements for modern https://.

4.1. Non-requirements

- Supporting network-specific DNS64 [RFC6147]
- Supporting other network-specific inferences from plaintext DNS queries
- Supporting insecure HTTP
- Supporting legacy HTTP versions

5. The HTTP Request

The URI scheme MUST be https.

The path SHOULD be "/.well-known/dns-query" but a different path can be used if the DNS API Client has prior knowledge about a DNS API

service on a different path at the origin being used. (See Section 8 for the registration of this in the well-known URI registry.) Using the well-known path allows automated discovery of a DNS API Service, and also helps contextualize DNS Query requests pushed over an active HTTP/2 connection.

A DNS API Client encodes the DNS query into the HTTP request using either the HTTP GET or POST methods.

When using the POST method, the DNS query is included as the message body of the HTTP request and the Content-Type request header indicates the media type of the message. POST-ed requests are smaller than their GET equivalents.

When using the GET method, the URI path MUST contain a query parameter of the form content-type=TTT and another of the form body=BBBB, where "TTT" is the media type of the format used for the body parameter, and "BBB" is the content of the body encoded with base64url [RFC4648]. Using the GET method is friendlier to many HTTP cache implementations.

The DNS API Client SHOULD include an HTTP "Accept:" request header to say what type of content can be understood in response. The client MUST be prepared to process "application/dns-udpwireformat" {{dnswire} responses but MAY process any other type it receives.

In order to maximize cache friendliness, DNS API clients using media formats that include DNS ID, such as application/dns-udpwireformat, should use a DNS ID of 0 in every DNS request. HTTP semantics correlate the request and response, thus eliminating the need for the ID in a media type such as application/dns-udpwireformat.

DNS API clients can use HTTP/2 padding and compression in the same way that other HTTP/2 clients use (or don’t use) them.

5.1. DNS Wire Format

The media type is "application/dns-udpwireformat". The body is the DNS on-the-wire format is defined in [RFC1035]. The body MUST be encoded with base64url [RFC4648]. Padding characters for base64url MUST NOT be included.

DNS API clients using the DNS wire format MAY have one or more EDNS(0) extensions [RFC6891] in the request.

5.2. Examples

For example, assume a DNS API server is following this specification on origin https://dnsserver.example.net/ and the well-known path. The DNS API client chooses to send its requests in appliation/dns-udpwirefomat but indicates it can parse replies in that format or as a JSON-based content type.

The examples uses HTTP/2 formatting from [RFC7540].

A query for the IN A records for "www.example.com" with recursion turned on using the GET method and a wireformat request would be:

```
:method = GET
:scheme = https
:authority = dnsserver.example.net
:path = /.well-known/dns-query?  (no CR)
        content-type=application/dns-udpwireformat&  (no CR)
        body=q80BAAABAAAAAAAAA3d3dwdleGFtcGxlA2NvbQAAAQAB
accept = application/dns-udpwireformat, application/simpledns+json
```

The same DNS query, using the POST method would be:

```
:method = POST
:scheme = https
:authority = dnsserver.example.net
:path = /.well-known/dns-query
accept = application/dns-udpwireformat, application/simpledns+json
content-type = application/dns-udpwireformat
content-length = 33

<33 bytes represented by the following hex encoding>
abcd 0100 0001 0000 0000 0000 0377 7777
0765 7861 6d70 6c65 0363 6f6d 0000 0100
01
```

6. The HTTP Response

Different response media types will provide more or less information from a DNS response. For example, one response type might include the information from the DNS header bytes while another might omit it. The amount and type of information that a media type gives is solely up to the format, and not defined in this protocol.

At the time this is published, the response types are works in progress. The only known response type is "application/dns-udpwireformat", but it is likely that at least one JSON-based response format will be defined in the future.

The DNS response for "application/dns-udpwireformat" in Section 5.1 MAY have one or more EDNS(0) extensions, depending on the extension definition of the extensions given in the DNS request.

Native HTTP methods are used to correlate requests and responses. Responses may be returned in a different temporal order than requests were made using the protocols native multi-streaming functionality.

In the HTTP responses, the HTTP cache headers SHOULD be set to expire at the same time as the shortest DNS TTL in the response. Because DNS provides only caching but not revalidation semantics, DNS over HTTP responses should not carry revalidation response headers (such as Last-Modified: or Etag:) or return 304 responses.

A DNS API Server MUST be able to process application/dns-udpwireformat request messages.

A DNS API Server SHOULD respond with HTTP status code 415 upon receiving a media type it is unable to process.

This document does not change the definition of any HTTP response codes or otherwise proscribe their use.

6.1. Example

This is an example response for a query for the IN A records for "www.example.com" with recursion turned on. The response bears one record with an address of 93.184.216.34 and a TTL of 128 seconds.

```
:status = 200
content-type = application/dns-udpwireformat
content-length = 64
cache-control = max-age=128

<64 bytes represented by the following hex encoding>
abcd 8180 0001 0001 0000 0000 0377 7777
0765 7861 6d70 6c65 0363 6f6d 0000 0100

0103 7777 7707 6578 616d 706c 6503 636f
6d00 0001 0001 0000 0080 0004 5db8 d822
```

7. HTTP Integration

In order to satisfy the security requirements of DNS over HTTPS, this protocol MUST use HTTP/2 [RFC7540] or its successors. HTTP/2 enforces a modern TLS profile necessary for achieving the security requirements of this protocol.

This protocol MUST be used with https scheme URI [RFC7230].

The messages in classic UDP based DNS [RFC1035] are inherently unordered and have low overhead. A competitive HTTP transport needs to support reordering, priority, parallelism, and header compression. For this additional reason, this protocol MUST use HTTP/2 [RFC7540] or its successors.

8. IANA Considerations

8.1. Registration of Well-Known URI

This specification registers a Well-Known URI [RFC5785]:

- URI Suffix: dns-query
- Change Controller: IETF
- Specification Document(s): [this specification]

8.2. Registration of application/dns-udpwireformat Media Type

To: ietf-types@iana.org
Subject: Registration of MIME media type
pplication/dns-udpwireformat

MIME media type name: application

MIME subtype name: dns-udpwireformat

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: This is a binary format. The contents are a DNS message as defined in RFC 1035. The format used here is for DNS over UDP, which is the format defined in the diagrams in RFC 1035.

Security considerations: The security considerations for carrying this data are the same for carrying DNS without encryption.

Interoperability considerations: None.

Published specification: This document.

Applications that use this media type:
Systems that want to exchange full DNS messages.

Additional information:

Magic number(s): n/a

File extension(s): n/a

Macintosh file type code(s): n/a

Person & email address to contact for further information:
Paul Hoffman, paul.hoffman@icann.org

Intended usage: COMMON

Restrictions on usage: n/a

Author: Paul Hoffman, paul.hoffman@icann.org

Change controller: IESG

9. Security Considerations

Running DNS over https:// relies on the security of the underlying HTTP connection. By requiring at least [RFC7540] levels of support for TLS this protocol expects to use current best practices for secure transport.

Session level encryption has well known weaknesses with respect to traffic analysis which might be particularly acute when dealing with DNS queries. Sections 10.6 (Compression) and 10.7 (Padding) of [RFC7540] provide some further advice on mitigations within an HTTP/2 context.

A server that is acting both as a normal web server and a DNS API server is in a position to choose which DNS names it forces a client to resolve (through its web service) and also be the one to answer those queries (through its DNS API service). An untrusted DNS API server can thus easily cause damage by poisoning a client's cache with names that the DNS API server chooses to poison. A client MUST NOT trust a DNS API server simply because it was discovered, or because the client was told to trust the DNS API server by an untrusted party. Instead, a client MUST only trust DNS API server that is configured as trustworthy.

10. Acknowledgments

Joe Hildebrand contributed lots of material for a different iteration of this document. Helpful early comments were given by Ben Schwartz and Mark Nottingham.

11. References

11.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://www.rfc-editor.org/info/rfc1035>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <http://www.rfc-editor.org/info/rfc4648>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <http://www.rfc-editor.org/info/rfc5785>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc7540>.

[RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <http://www.rfc-editor.org/info/rfc7858>.

11.2. Informative References

[CORS] W3C, "Cross-Origin Resource Sharing", 2014, <https://www.w3.org/TR/cors/>.

[I-D.ietf-dnsop-dns-wireformat-http] Song, L., Vixie, P., Kerr, S., and R. Wan, "DNS wire-format over HTTP", draft-ietf-dnsop-dns-wireformat-http-01 (work in progress), March 2017.

[RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, DOI 10.17487/RFC6147, April 2011, <http://www.rfc-editor.org/info/rfc6147>.

[RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <http://www.rfc-editor.org/info/rfc6891>.

Appendix A. Previous Work on DNS over HTTP or in Other Formats

The following is an incomplete list of earlier work that related to DNS over HTTP/1 or representing DNS data in other formats.

The list includes links to the tools.ietf.org site (because these documents are all expired) and web sites of software.

- https://tools.ietf.org/html/draft-mohan-dns-query-xml
- https://tools.ietf.org/html/draft-daley-dnsxml
- https://tools.ietf.org/html/draft-dulaunoy-dnsop-passive-dns-cof
- https://tools.ietf.org/html/draft-bortzmeyer-dns-json
- https://www.nlnetlabs.nl/projects/dnssec-trigger/

Authors' Addresses

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

Patrick McManus
Mozilla

Email: pmcmanus@mozilla.com

The Dispatch Working Group
Internet-Draft
Intended status: Informational
Expires: January 4, 2018

R. Huang
H. Zheng
R. Even
Huawei

July 03, 2017

Video Delivery in Hybrid Network
draft-huang-dispatch-hybrid-video-delivery-00

Abstract

The industry trend of delivering video service is moving towards all IP solutions. However, there exit multiple incompatible platforms for video distribution. This document explores the existing video delivery technologies and analyses the challenges of unifying those technologies.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 4, 2018.

Copyright Notice

Table of Contents

1. Introduction

Video content delivery is now the major bandwidth usage over the Internet. Globally, IP video traffic will be 82 percent of all IP traffic (both business and consumer) by 2020, up from 70 percent in 2015. 4K Ultra HD technology is by itself a very new trend in the overall electronics landscape, and the impact of it is growing month by month. More content is accessible, in more formats, on more devices, for more people than ever before. Content providers and broadcasters are embracing multi-platform to attract more audiences. For example, IPTV providers not just provide video services on fix network but also consider to start the services for mobile accessed users; Traditional broadcasters not just provide services over cable or satellite, but also consider to start the services over IP network. How to transmit video traffic efficiently over these mutli-platforms poses challenges to these service providers.

This document explores the existing video delivery technologies and analyses these challenges.

2. Terminology

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

3. Multi-platform for Video distribution

The same content could be delivered in different networks including broadband, mobile, satellite, cable and terrestrial. And the receiving devices can be all kinds of STBs, mobile phones, tablets and PCs. This is shown in Figure 1.

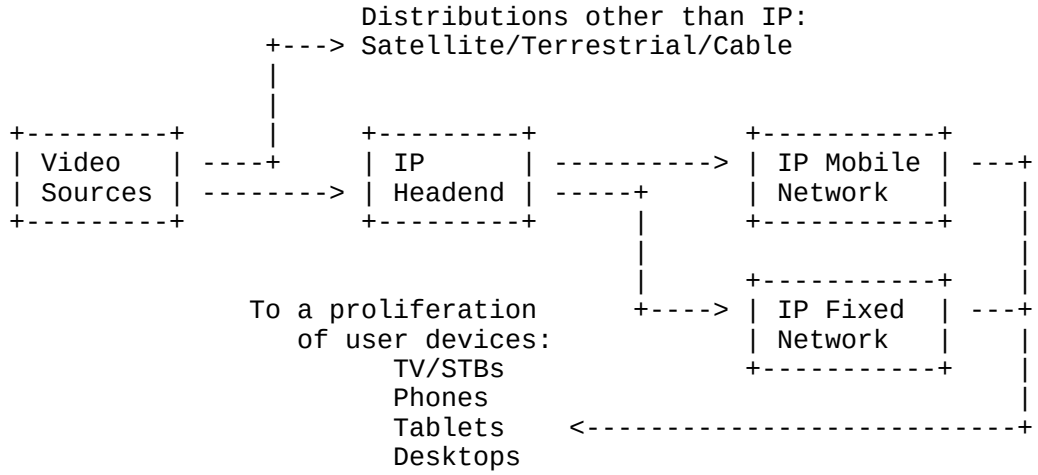

Figure 1: Multi-platform Distribution for Video

4. Looking into the Protocols

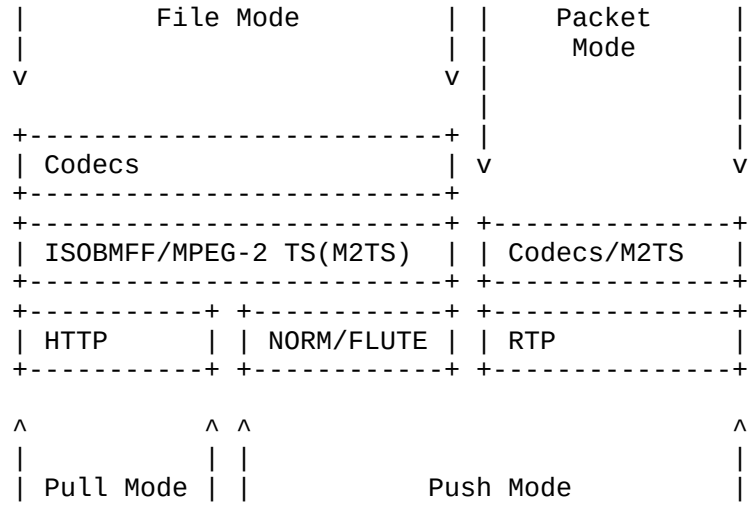

Figure 2: Video Delivery Protocol Stacks

Today, there exist many diverged video delivery protocol stacks, as listed in Figure 2. Looking bottom-up, from the angle of transmission methods, the protocol stacks can be categorized into two modes: "Pull Mode" and "Push Mode". In "Pull Mode", client takes the initiative and pulls content from server proactively. Typical "Pull Mode" methods include HTTP progressive download and HTTP Adaptive

Stream (e.g. HLS and DASH). On the other hand, "Push Mode" operations are more server oriented. After session has been established, server controls the delivery by intentionally pushing content to client. Typical "Pull Mode" transmissions include IPTV and other multicast based methods.

Another way to look at the protocol stacks is from the top-down angle, regarding how media are prepared before transmission. Traditional media delivery utilizes "Packet Mode", in which media is packetized with regard to their internal structure, so the resulting packets are optimized for transport and more loss tolerant. An example is transporting H.264 encoded video directly over RTP. Unlike "Packet Mode", segmented media grows more popular as they are adopted by HTTP Adaptive Streaming. Segmented media are referred to as "File Mode" in Figure 2, for the fact that media segments are seen as plain files, and described by additional manifests.

The divergence in the protocol stacks has brought several issues, as the bottom-up angle and the top-down angel do not align with each other ("File Mode" and "Push Mode" are overlapping):

- "File Mode" is not quite suitable to be used in "Push Mode", as the transports lack timing information.
- "File Mode" is very difficult to be converted into "Packet Mode", and thus cannot be transported using unreliable protocols such as RTP.

The divergence increases the overall complexity of video delivery. The next section analyzes the impact introduced by the complexity.

5. Impact of Diversity on IP distribution

Currently the IP Headend (as in Figure 1) is unduly complicated by the diversity on IP distribution, as illustrated below:

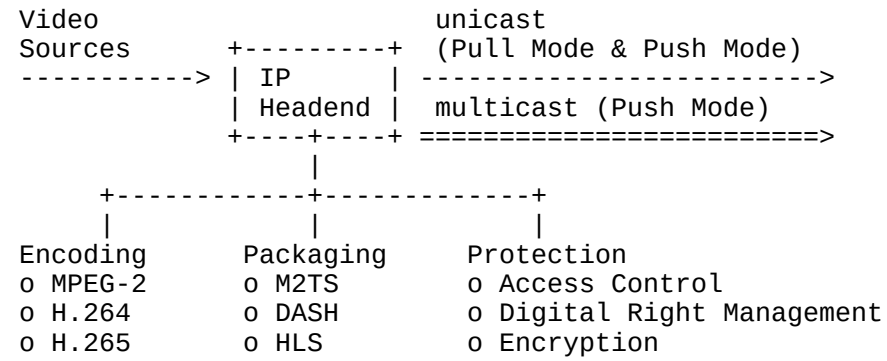

Figure 3: Complicated IP Headend

It is a tedious job for the IP Headend to encode the same content into different variants using different media profiles, prepare them in several types of packaging, and apply different protection mechanisms before the variants are served. The consequence is increased cost in design, deployment, test and operation.

The situation is further complicated by the diversity of network delivery mechanisms and content forms. Unicast delivery supports "Pull Mode" and "Push Mode", whereas multicast delivery only supports "Push Mode". Each delivery mechanism uses different transport protocols and support different content forms. "Pull Mode" supports "File Mode" content, and "Push Mode" supports content in both "File Mode" and "Push Mode".

"File Mode" content is usually served in "Pull mode". However, it can also be served in "Push Mode" by using reliable multicast technologies (e.g. FLUTE, NORM). Serving "File Mode" content with "Push Mode" delivery would increase delay, as the reliability mechanisms imply using retransmission to recover lost data. It has impact on applications that require low-delay transport, for example, live video or virtual reality.

If an application can tolerate a level of packet loss, then it is possible for the application to transform content from "File Mode" into "Packet mode", and transfer more efficiently in "Push Mode". An example would be to transform HLS media segments of MPEG-2 TS format into RTP packets, and multicast those RTP packets carrying MPEG-2 TS content to endpoints. However, this is only possible if the application is authorized to access the content and do the transformation. It is usually not the case in real-life scenarios. In order to protect contents, such transformation is not allowed in delivery by content providers.

There have been efforts to provide convergence for this diversified situation. New media packaging formats such as MMT, CMAF are proposed by MPEG that can packetize the media in application layer. So the same packaged media content can support both "File Mode" and "Packet Mode". To support the new packaging formats, maybe a content agnostic transport protocol should be developed here in IETF.

6. Security Considerations

TBD.

7. IANA Considerations

None.

8. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

Authors' Addresses

Rachel Huang
Huawei

Email: rachel.huang@huawei.com

Hui Zheng
Huawei

Email: marvin.zhenghui@huawei.com

Roni Even
Huawei

Email: roni.even@huawei.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 4, 2018

D. Thaler
Microsoft

July 3, 2017

Using URIs With Multiple Transport Stacks
draft-thaler-appsawg-multi-transport-uris-01

Abstract

Many Uniform Resource Identifiers (URIs) today have some mechanism to resolve them to one or more specific endpoints where that resource is available. This document discusses issues that arise when the same resource can be reached over multiple protocol stacks, and discusses various approaches that have been used or discussed, and the tradeoffs between them. Such issues are important to consider when defining new URI schemes and resolution mechanisms.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 4, 2018.

Table of Contents

1. Introduction

For Uniform Resource Identifier (URI) schemes that function as locators, [RFC3986] explains that URI "resolution" is the process of determining an access mechanism and the appropriate parameters necessary to deference a URI; this resolution may require several iterations. To use that access mechanism to perform an action on the URI's resource is to "dereference" the URI.

The specific details vary by URI scheme and hence are up to each URI scheme definition to specify. Requirements for URI scheme definitions are covered in [RFC3986], [RFC7320], and [RFC7595]. RFC 7595 section 3.3 states:

> For schemes that function as locators, it is important that the mechanism of resource location be clearly defined.

Closely related to the concept of resolving a URI to a resource that may have multiple ways to reach it, is the concept of "equivalence". [RFC3986] section 6.1 states:

> Even though it is possible to determine that two URIs are equivalent, URI comparison is not sufficient to determine whether two URIs identify different resources. For example, an owner of two different domain names could decide to serve the same resource from both, resulting in two different URIs. Therefore, comparison methods are designed to minimize false negatives while strictly avoiding false positives.

Thus, it is possible that two distinct URIs refer to the same resource. The goal, as RFC 3986 stated above, is simply to "minimize" such cases, but such minimization often comes at a cost. For example, for many URIs schemes, a DNS name can be used in the authority component rather than using several URIs that differ only in IP address literal, with the cost being a dependency on DNS name resolution and the potential latency and traffic involved.

As another example, [RFC5630] section 4.1 states:

> SIP and SIPS URIs that are identical except for the scheme itself (e.g., sip:alice@example.com and sips:alice@example.com) refer to the same resource. This requirement is implicit in [RFC3261], Section 19.1, which states that "any resource described by a SIP URI can be 'upgraded' to a SIPS URI by just changing the scheme, if it is desired to communicate with that resource securely". This does not mean that the SIPS URI will necessarily be reachable, in particular, if the proxy cannot establish a secure connection to a client or another proxy. This does not suggest either that proxies would arbitrarily "upgrade" SIP URIs to SIPS URIs when forwarding a request (see Section 5.3). Rather, it means that when a resource is addressable with SIP, it will also be addressable with SIPS.

Thus, the same resource might be identified by multiple URIs that differ only in URI scheme, or authority component, or path (e.g., using ".." resolution).

For URIs used in the World Wide Web, Section 2.3.1 of "Architecture of the World Wide Web" [AWWW] further discusses such aliasing, explaining that links to a resource increase the value of that resource, and multiple URIs for it interfere with such valuation, and also makes it difficult to correlate two sources as pointing to the same resource via differing aliases. Thus to maximize the benefit to the Web, URI aliases should be minimized.

2. Problem Statement

Besides specifying one or more URI scheme names to be used and the syntax for each (e.g., what the authority component contains), there are two issues a URI scheme definer must deal with when multiple transports are available for accessing a given resource:

1. Specifying how the set of transport endpoint identifiers (e.g., TCP and UDP port numbers) for a given URI can be discovered by an entity wishing to resolve it, and

2. Specifying how an appropriate transport endpoint can be selected for use, from among the discovered set.

At a high level, these issues are equivalent to those arising when multiple IP addresses are available for the same resource. However, in general, there may be multiple layers in a transport stack, each with their own identifiers, so the problems are compounded when multiple choices exist at each of multiple layers below the application-layer protocol itself.

3. Transport endpoint discovery

A client wishing to access a resource needs to know, for each layer in the transport stack, what protocol(s) to use, and what identifier(s) are needed by each such protocol. There are several possible approaches to transport endpoint identifier discovery, which we cover in the following sections. For simplicity, we will discuss them as if the same approach is used for both types of information, but it is important to remember that a URI scheme could specify discovery of the set of transport protocols via one approach, and discovery of the identifier(s) for each transport protocol via another approach.

3.1. Specified by the URI scheme specification

In this approach, every resource is assumed to use the exact same set of transport protocols (i.e., stacks of protocols above the network layer) and identifiers. The identifiers can be IANA assigned and specified as part of the URI scheme or protocol specification. For example, TFTP only supports UDP port 69, and so no port number is permitted in a tftp URI.

If support for a new transport protocol is later added under a protocol with a given URI scheme, different entities may thus have different hard-coded assumptions about the set of possible transport protocols, which just pushes the rest of the burden to the problem of selection among the known set (see Section 4).

A disadvantage of this approach for many use cases is that it does not allow for non-default configurations such as custom ports.

3.2. Passed in one URI

For single-transport protocols, a common mechanism is to specify a default port for the URI scheme, and to allow putting a non-default port number in the URI authority component.

For multi-transport protocols, historically it was sometimes assumed that multiple transport protocols (e.g., UDP and TCP) would use the same port number, so specifying a single number would also be sufficient for multiple transports. When port numbers appear in URIs, they are not the default ports that might be IANA-assigned (since default ports should be omitted from the URI per [RFC3986] section 3.2.3), but instead are either statically chosen by the server application, or are ephemeral ports dynamically allocated on the server hosting the resource. In most TCP/IP stacks, ephemeral ports used by UDP endpoints have no relationship to ephemeral ports used by TCP endpoints in the same application and so it cannot be guaranteed that the port numbers are the same. For example, port 51000 might be allocated to one application for UDP, and a different application for TCP.

Since 2011, this same issue can also occur with IANA-assigned ports, especially if support for a given transport protocol is added at a later time. [RFC6335] section 7.2 explains:

> Effective with the publication of this document, IANA will begin assigning port numbers for only those transport protocols explicitly included in an assignment request. This ends the long-standing practice of automatically assigning a port number to an application for both TCP and UDP, even if the request is for only one of these transport protocols.

Thus, for most URI schemes, a port number appearing in a URI authority component must be specified as being in a specific transport-layer protocol’s numbering space since its value for a given resource might differ by transport protocol. If a URI scheme wishes for the port number in URI authority component to be able to apply to multiple transport protocols, the URI scheme would typically have to assume static configuration on servers; this may be acceptable in some circumstances and unacceptable in others.

A common solution in non-URI contexts is to use a service name rather than a literal port number, and allow the service name to be resolved to the relevant transport-layer identifier. Indeed, [RFC6335] section 3 says:

> Because the port number space is finite (and therefore conservation is an important goal), the alternative of using service names instead of port numbers is RECOMMENDED whenever possible.

Unfortunately, it is not possible to follow this recommendation with the port field in URI authority component, since the URI syntax only allows integers in the port field.

For new URI schemes, it may be possible in some cases to place a service name in the host field, such as "_myservice._tcp.example.org" as would be used with a DNS SRV record [RFC2782]. That example still specifies only a single transport protocol stack ("_tcp") however, rather than a list of supported stacks.

Another limitation of service names is that they are currently limited only to TCP, UDP, SCTP, and DCCP, and so cannot be used with other layers (e.g., websockets) or protocols. Thus, a URI scheme for a protocol that supports both, say, websockets and raw TCP as possible transports for resource access, cannot use a service name as a common identifier for transport-layer endpoint resolution.

It is usually also undesirable to put transport-layer endpoint information (the list of supported transport protocols or the identifier(s) used with the transport protocols) in the path or query components for two reasons. First, those components are typically passed over the wire to the server when accessing a resource, which only consumes extra bandwidth with no benefit. Second, if the transport-layer identifiers might change over the lifetime of the resource, then the URI would need to change even if the change did not affect the actual endpoint chosen by the client. Such a change would negatively affect equivalence with the previous URI, e.g., resulting in cache misses.

Thus, an advantage of this approach is that it can work without any dependency on other protocols or deployment of servers needed for resolution, and a disadvantage is that putting information about multiple transport-layer endpoints anywhere in the same URI could make for a very long URI that might have issues with certain software, or have bandwidth or storage issues.

3.3. Use separate URI for each transport endpoint

In this approach, one must simply accept the fact that multiple URIs might refer to the same resource as RFC 3986 already allows. This is similar to using a set of URIs that differ only in IP address literal, for a case when the resource server is not resolvable via a protocol such as DNS or SIP.

The obvious disadvantage is that there are multiple URIs for the same resource. Another potential disadvantage for some more complex use cases where there are multiple layers of the transport stack, is that it may be difficult or impossible to express all the identifiers in an entire stack of protocols in one URI.

For cases where there are multiple transport protocols but only one such layer, this approach results in needing to identify a single

transport protocol per URI. As discussed in Section 3.2, this often cannot be put in the authority component and is undesirable to put in the path or query component. As a result, such cases involve specifying a separate URI scheme per transport. For example, "sip" and "sips" do this. The CoRE WG also proposed this approach for CoAP with "coap", "coaps", "coap+tcp", "coaps+tcp", etc.

3.4. Use another mechanism for discovery

In this approach, a URI scheme definer would specify a mechanism whereby transport stack identifiers can be resolved for a given URI. If multiple layers exist, then such resolution might involve a resolution step for each layer.

DNS records (e.g., SRV records) provide one potential mechanism that can be used to discover a set of supported transports and their associated identifiers. Other types of directories might be usable in other cases. For example, HTTP now provides an "Alt-Svc" [RFC7838] mechanism that can discover alternate transport endpoints for the same HTTP URI.

One challenge in many cases is defining a common mechanism that could discover identifiers for different transport protocols. For example, websockets use URIs and TCP uses port numbers (and there is currently no URI scheme for TCP itself), and so the syntax of such identifiers may differ if an application layer protocol could use both TCP and websockets.

The advantage of requiring a separate resolution mechanism is that the resource URI itself can be kept short and simple. The downside is extra complexity in both clients and servers, and potentially extra specification work for the URI scheme definer, and the possible additional deployment burden of provisioning and operating extra protocols or servers to facilitate such resolution.

In some contexts, it might also be feasible to discover the additional identifiers using the same mechanism used to discover the URI itself, perhaps even in the same message.

4. Transport endpoint selection

The URI scheme must specify the mechanism for choosing among transport protocol stacks, such as specifying at least one that is mandatory to implement and an algorithm for trying possible transport stacks in some order until one works.

This problem is similar to that of choosing among multiple discovered IP addresses for the same transport stack, and two common solutions

are used today in that context. One category of algorithm is to sort the choices according to some criteria, and then to try them in order of preference. For example, SRV records provide a priority and weight for each transport endpoint that can be used to sort them, and [RFC6724] provides an algorithm for sorting destination IP addresses.

Another category of such algorithms is called "Happy Eyeballs" [RFC6555] where multiple possibilities are attempted in parallel (possibly with some delay added before starting non-preferred choices) and keeping the first one that successfully connects. The advantage is faster connection when a non-preferred choice is needed, and the disadvantages are extra complexity in the client, extra traffic on the network, and extra connections at the server if multiple parallel attempts succeed.

As noted earlier, when multiple layers exist in the transport stack, the number of possible permutations might be large in some cases, and so a mechanism must be cognizant of that.

5. IANA Considerations

This document has no actions for IANA.

6. Security Considerations

The security considerations in section 3.7 of [RFC7595] and section 7 of [RFC3986] apply. [RFC6943] also discusses security considerations with determining equivalence, and section 3.1.4 of that document is relevant to resolution. This document does not raise additional security issues.

7. Acknowledgements

Thanks to Graham Klyne, Alexey Melnikov, and Gabriel Montenegro for helpful suggestions on this document.

8. Informative References

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <http://www.rfc-editor.org/info/rfc2782>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC5630] Audet, F., "The Use of the SIPS URI Scheme in the Session Initiation Protocol (SIP)", RFC 5630, DOI 10.17487/RFC5630, October 2009, <http://www.rfc-editor.org/info/rfc5630>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <http://www.rfc-editor.org/info/rfc6335>.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April 2012, <http://www.rfc-editor.org/info/rfc6555>.

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <http://www.rfc-editor.org/info/rfc6724>.

[RFC6943] Thaler, D., Ed., "Issues in Identifier Comparison for Security Purposes", RFC 6943, DOI 10.17487/RFC6943, May 2013, <http://www.rfc-editor.org/info/rfc6943>.

[RFC7320] Nottingham, M., "URI Design and Ownership", BCP 190, RFC 7320, DOI 10.17487/RFC7320, July 2014, <http://www.rfc-editor.org/info/rfc7320>.

[RFC7595] Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, DOI 10.17487/RFC7595, June 2015, <http://www.rfc-editor.org/info/rfc7595>.

[RFC7838] Nottingham, M., McManus, P., and J. Reschke, "HTTP Alternative Services", RFC 7838, DOI 10.17487/RFC7838, April 2016, <http://www.rfc-editor.org/info/rfc7838>.

[AWWW] Jacobs, I. and N. Walsh, "Architecture of the World Wide Web, Volume One", December 2004, <http://www.w3.org/TR/webarch>.

Author’s Address

Dave Thaler
Microsoft
One Microsoft Way
Redmond, WA 98052
USA

Email: dthaler@microsoft.com