

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 6, 2018

D. Thaler
Microsoft
March 5, 2018

Using URIs With Multiple Protocol Stacks
draft-thaler-appsawg-multi-transport-uris-02

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 <https://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on September 6, 2018.

Copyright Notice

Copyright (c) 2018 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 (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in [Section 4.e](#) of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
2.	Problem Statement	4
3.	Protocol endpoint discovery	4
3.1.	Specified by the URI scheme specification	5
3.2.	Passed in one URI	5
3.3.	Use separate URI for each transport endpoint	7
3.4.	Use another mechanism for discovery	7
4.	Transport endpoint selection	8
5.	IANA Considerations	9
6.	Security Considerations	9
7.	Acknowledgements	9
8.	Informative References	9
	Author's Address	10

[1.](#) Introduction

For Uniform Resource Identifier (URI) schemes that function as locators (historically called "URLs"), [\[RFC3986\]](#) 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.

Similarly, the same resource might be identified using both "http" and "https", and indeed a commonly followed rule (section 4.1.3 of [\[USWP\]](#)) is that the URI scheme sets expectations for integrity of access, such that separate integrity levels result in separate URI schemes.

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.

See "URI Schemes and Web Protocols" [[USWP](#)] for additional discussion on the relationship between URI schemes and protocols in a web context, although that document has no official standing and there is

a history of difficulty in reaching consensus on the connection between URI schemes and protocols. [[Noah](#)]

[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 protocol stacks are available for accessing a given resource:

1. Specifying how the set of protocol 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 protocol 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 (e.g., some application-layer protocol over WebSockets over TCP), each with its own identifiers, so the problems are compounded when multiple choices exist at each of multiple layers below the application-layer protocol itself.

Thus, when we use the term "protocol stack" in this document, we typically mean the stack of protocols below the application-layer protocol associated with the URI scheme, and above the network layer. However, [[USWP](#)] also discusses the possibility ("Approach 2") that

multiple application-layer protocols might share the same URI scheme, in which case the "protocol stack" also includes the application-layer protocols to select from.

[3.](#) Protocol endpoint discovery

A client wishing to access a resource needs to know, for each layer in the protocol stack, what protocol(s) can be used, and what identifier(s) are needed by each such protocol. There are several possible approaches to 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 protocols via one approach, and discovery of the identifier(s) for each 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 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 server 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 the 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, as do "http" and "https". [RFC 8323](#) [[RFC8323](#)] also follows 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, and the identifiers would come in a form that may not be expressed as a 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. Another example mentioned in [[USWP](#)] is where

the protocol to use is identified by a media type value.

One challenge in many cases is defining a common mechanism that could discover identifiers for different transport protocols for the same resource. 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 downsides are extra complexity in both clients and servers, potentially extra specification work for the URI scheme definer, the possible additional deployment burden of provisioning and operating extra protocols or servers to facilitate such resolution, and any additional bandwidth or latency of doing the resolution.

In some contexts, it might 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 should 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. The URI scheme might even leave it up to the client implementation or client configuration options as suggested in Approach 2 of [[USWP](#)].

The endpoint selection 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 responds successfully. 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, <<https://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, <<https://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, <<https://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, <<https://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, <<https://www.rfc-editor.org/info/rfc6555>>.

Thaler

Expires September 6, 2018

[Page 9]

Internet-Draft

Multi-Transport URIs

March 2018

- [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, <<https://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, <<https://www.rfc-editor.org/info/rfc6943>>.
- [RFC7320] Nottingham, M., "URI Design and Ownership", [BCP 190](#), [RFC 7320](#), DOI 10.17487/RFC7320, July 2014, <<https://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, <<https://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, <<https://www.rfc-editor.org/info/rfc7838>>.
- [RFC8323] Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, Ed., "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", [RFC 8323](#), DOI 10.17487/RFC8323, February 2018, <<https://www.rfc-editor.org/info/rfc8323>>.
- [Awww] Jacobs, I. and N. Walsh, "Architecture of the World Wide Web, Volume One", December 2004, <<http://www.w3.org/TR/webarch>>.
- [USWP] Mendelsohn, N., "URI Schemes and Web Protocols", November 2005, <<http://www.w3.org/2001/tag/doc/SchemeProtocols.html>>.
- [Noah] Mendelsohn, N., "Email from Noah Mendelsohn to the URI-Review mailing list", July 2017, <<https://www.ietf.org/mail-archive/web/uri-review/current/msg01919.html>>.

Author's Address

Thaler

Expires September 6, 2018

[Page 10]

Internet-Draft

Multi-Transport URIs

March 2018

Dave Thaler
Microsoft
One Microsoft Way
Redmond, WA 98052
USA

Email: dthaler@microsoft.com

Thaler

Expires September 6, 2018

[Page 11]