Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 15, 2014

M. Nottingham
Akamai
P. McManus
Mozilla

February 11, 2014

HTTP Alternate Services
draft-nottingham-httpbis-alt-svc-03

Abstract

This document proposes a number of changes to HTTP, centered around "alternate services", which allow an origin's resources to be available at a separate network location, possibly accessed with a different protocol configuration.

Status of this Memo

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

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

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

This Internet-Draft will expire on August 15, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction

[I-D.ietf-httpbis-http2] specifies a few ways to negotiate the use of HTTP/2 without changing existing URIs. However, several deficiencies in using the "upgrade dance" for "http://" URIs have become apparent. While that mechanism is still being investigated, some have expressed interest in an alternate approach.

Furthermore, some implementers have expressed a strong desire utilize HTTP/2 only in conjunction with TLS. Alternate-Services provides a potential mechanism for achieving that for "http://" URIs; see [I-D.nottingham-http2-encryption] for details.

Finally, HTTP/2 is designed to have longer-lived, fewer and more active TCP connections. While these properties are generally "friendlier" for the network, they can cause problems for servers that currently exploit the short-lived flow characteristics of HTTP/1.x for load balancing, session affinity and maintaining locality to the user.

This document explores these use cases in Section 2, and makes proposals to address them in Section 3.

1.1. Notational Conventions

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

This document uses the Augmented BNF defined in [RFC5234] along with the "OWS", "DIGIT", "parameter", "uri-host", "port" and "delta-second" rules from [I-D.ietf-httpbis-p1-messaging], and uses the "#rule" extension defined in Section 7 of that document.

2. Use Cases for Alternate Services

This section details the use cases for Alternate Services, identifying the proposals that would need to be adopted to meet each one.

2.1. Upgrading HTTP/1

The first use case for Alternate Services is upgrading a client from HTTP/1 to HTTP/2 for http:// URIs (i.e., without the use of SSL or TLS).

While HTTP/2 defines how to use the Upgrade header "dance" to do

this, it might not always be suitable, since it involves blocking the connection until the upgrade succeeds. Since HTTP/2 is focused on improving performance, this is not desirable.

Furthermore, using Upgrade requires the server that supports HTTP/2 to be on the same ip:port tuple as the server supporting HTTP/1; this can cause deployment issues, as well as operational issues with devices that assume that all traffic on port 80 will be HTTP/1.

Therefore, a means of indicating that a different, new connection can be used for HTTP/2 is desirable; this would allow the client to continue using the HTTP/1 connection until the new connection is established. It also simplifies deployment considerations by not requiring HTTP/1 and HTTP/2 to be "spoken" on the same port, and allows issues on port 80 to be avoided.

This use case can be met if Section 3.1 and Section 3.2 are accepted. It can also be met if Section 3.1 is used with a different discovery mechanism (e.g., DNS-based).

2.2. Using TLS with http:// URIs

As discussed in [I-D.nottingham-http2-encryption], it might be desirable to "opportunistically" use TLS when accessing a HTTP URI.

This case can also be met by Section 3.1 and Section 3.6 with Section 3.2 for HTTP/1, and Section 3.1 and Section 3.6 with Section 3.2 or Section 3.3 for HTTP/2, with the possible addition of Section 3.4. Section 3.5 might also be useful for handling errors in this use case.

2.3. Mitigating Load Asymmetry

HTTP/2 fundamentally changes how HTTP uses TCP. While this has many benefits, it also disrupts many server-side practices that take advantage of HTTP/1's shorter flows.

In particular, load balancing among a pool of servers is often used, either with DNS ("global" load balancing), or a device (hardware or software) that dispatches requests locally.

HTTP/1's short flows aid in this, because it is easy to shift traffic among servers when one becomes overloaded or unavailable. However, in HTTP/2, this is more difficult, due to the protocol's longer flows.

As a result, a mechanism for re-directing requests for an origin or set of origins without making this apparent to the application is

desirable.

This use case can be met if Section 3.1 and Section 3.3 are accepted; servers can redirect clients to alternate services as appropriate. Section 3.5 might also be useful for handling errors in this use case.

2.4. Segmenting Clients that Support TLS SNI

TLS Server Name Indication (SNI) [RFC6066] was introduced to avoid a requirement for a 1:1 mapping between origin hostnames and IP addresses (in light of IPv4 address exhaustion), in a manner similar to the Host header in HTTP/1.

However, there are still clients in common use that do not send SNI. As a result, servers have no way to take practical advantage of the extension, because there is no way to segment those clients that support SNI from those that do not.

As a result, they need to build their infrastructure as if SNI did not exist.

This use case can be met if Section 3.1 and Section 3.3 are accepted; servers can advertise an alternate service and direct clients that support SNI and HTTP/2 to the optimal server, while still maintaining a smaller set of legacy servers for those clients that do not support SNI (since HTTP/2 requires SNI support when TLS is in use).

3. Proposals

This section enumerates proposals made to meet the use cases in Section 2. Note that they all need not be accepted together, depending on the use cases that are determined as in-scope.

3.1. Proposal: Alternate Services

NOTE: This section can be incorporated into HTTP/2 directly, or it can be published as a standalone specification. However, if Section 3.3 is accepted, it will need to be included or referenced from the spec, since frame type extensibility has been ruled out.

This specification defines a new concept in HTTP, the "alternate service." When an origin (see [RFC6454] has resources are accessible through a different protocol / host / port combination, it is said to have an alternate service.

An alternate service can be used to interact with the resources on an

origin server at a separate location on the network, possibly using a different protocol configuration.

For example, an origin:

("http", "www.example.com", "80")

might declare that its resources are also accessible at the alternate service:

("h2", "new.example.com", "81")

By their nature, alternate services are explicitly at the granularity of an origin; i.e., they cannot be selectively applied to resources within an origin.

Alternate services do not replace or change the origin for any given resource; in general, they are not visible to the software "above" the access mechanism. The alternate service is essentially alternate routing information that can also be used to reach the origin in the same way that DNS CNAME or SRV records define routing information at the name resolution level. Each origin maps to a set of these routes - the default route is derived from origin itself and the other routes are introduced based on alternate-protocol information.

Furthermore, it is important to note that the first member of an alternate service tuple is different from the "scheme" component of an origin; it is more specific, identifying not only the major version of the protocol being used, but potentially communication options for that protocol.

This means that clients using an alternate service will change the host, port and protocol that they are using to fetch resources, but these changes MUST NOT be propagated to the application that is using HTTP; from that standpoint, the URI being accessed and all information derived from it (scheme, host, port) are the same as before.

Importantly, this includes its security context; in particular, when TLS [RFC5246] is in use, the alternate server will need to present a certificate for the origin's host name, not that of the alternate. Likewise, the Host header is still derived from the origin, not the alternate service (just as it would if a CNAME were being used).

The changes MAY, however, be made visible in debugging tools, consoles, etc.

Formally, an alternate service is identified by the combination of:

- An ALPN protocol, as per [I-D.ietf-tls-applayerprotoneg]
- A host, as per [RFC3986]
- A port, as per [RFC3986]

Additionally, each alternate service MUST have:

- A freshness lifetime, expressed in seconds; see Section 3.1.2

There are many ways that a client could discover the alternate service(s) associated with an origin.

3.1.1. Host Authentication

Clients MUST NOT use alternate services with a host other than the origin's without strong server authentication; this mitigates the attack described in Section 4.2. One way to achieve this is for the alternate to use TLS with a certificate that is valid for that origin.

For example, if the origin's host is "www.example.com" and an alternate is offered on "other.example.com" with the "h2t" protocol, and the certificate offered is valid for "www.example.com", the client can use the alternate. However, if "other.example.com" is offered with the "h2" protocol, the client cannot use it, because there is no mechanism in that protocol to establish strong server authentication.

Furthermore, this means that the HTTP Host header and the SNI information provided in TLS by the client will be that of the origin, not the alternate.

3.1.2. Alternate Service Caching

Mechanisms for discovering alternate services can associate a freshness lifetime with them; for example, the Alt-Svc header field uses the "ma" parameter.

Clients MAY choose to use an alternate service instead of the origin at any time when it is considered fresh; see Section 3.1.4 for specific recommendations.

Clients with existing connections to alternate services are not required to fall back to the origin when its freshness lifetime ends; i.e., the caching mechanism is intended for limiting how long an alternate service can be used for establishing new requests, not limiting the use of existing ones.

To mitigate risks associated with caching compromised values (see

Section 4.2 for details), user agents SHOULD examine cached alternate services when they detect a change in network configuration, and remove any that could be compromised (for example, those whose association with the trust root is questionable). UAs that do not have a means of detecting network changes SHOULD place an upper bound on their lifetime.

3.1.3. Requiring Server Name Indication

A client must only use a TLS-based alternate service if the client also supports TLS Server Name Indication (SNI) [RFC6066]. This supports the conservation of IP addresses on the alternate service host.

3.1.4. Using Alternate Services

By their nature, alternate services are optional; clients are not required to use them. However, it is advantageous for clients to behave in a predictable way when they are used by servers (e.g., for load balancing).

Therefore, if a client becomes aware of an alternate service, the client SHOULD use that alternate service for all requests to the associated origin as soon as it is available, provided that the security properties of the alternate service protocol are desirable, as compared to the existing connection.

The client is not required to block requests; the origin's connection can be used until the alternate connection is established. However, if the security properties of the existing connection are weak (e.g. cleartext HTTP/1.1) then it might make sense to block until the new connection is fully available in order to avoid information leakage.

Furthermore, if the connection to the alternate service fails or is unresponsive, the client MAY fall back to using the origin. Note, however, that this could be the basis of a downgrade attack, thus losing any enhanced security properties of the alternate service.

3.2. Proposal: The Alt-Svc HTTP Header Field

NOTE: Because this header is mostly defined for use in HTTP/1, it is most likely most appropriate to put it in a separate specification. However, it will need to reference Section 3.1, wherever that is specified.

A HTTP(S) origin server can advertise the availability of alternate services (see Section 3.1) to clients by adding an Alt-Svc header field to responses.

```
Alt-Svc     = 1#( alternate *( OWS ";" OWS parameter ) )
alternate   = <"> protocol-id <"> "=" port
protocol-id = <ALPN protocol identifier>
```

For example:

```
Alt-Svc: http2=8000
```

This indicates that the "http2" protocol on the same host using the indicated port (in this case, 8000).

Alt-Svc does not allow advertisement of alternate services on other hosts, to protect against various header-based attacks.

It can, however, have multiple values:

```
Alt-Svc: "h2"=8000, "h2t"=443
```

The value(s) advertised by Alt-Svc can be used by clients to open a new connection to one or more alternate services immediately, or simultaneously with subsequent requests on the same connection.

Intermediaries MUST NOT change or append Alt-Svc values.

Finally, note that while it may be technically possible to put content other than printable ASCII in a HTTP header, some implementations only support ASCII (or a superset of it) in header field values. Therefore, this field SHOULD NOT be used to convey protocol identifiers that are not printable ASCII, or those that contain quote characters.

3.2.1. Caching Alt-Svc Header Field Values

When an alternate service is advertised using Alt-Svc, it is considered fresh for 24 hours from generation of the message. This can be modified with the 'ma' (max-age) parameter;

```
Alt-Svc: "h2t"=443;ma=3600
```

which indicates the number of seconds since the response was generated the alternate service is considered fresh for.

```
ma = delta-seconds
```

See [I-D.ietf-httpbis-p6-cache] Section 4.2.3 for details of determining response age. For example, a response:

```
HTTP/1.1 200 OK
Content-Type: text/html
Cache-Control: 600
Age: 30
Alt-Svc: "h2"=8000; ma=60
```

indicates that an alternate service is available and usable for the next 60 seconds. However, the response has already been cached for 30 seconds (as per the Age header field value), so therefore the alternate service is only fresh for the 30 seconds from when this response was received, minus estimated transit time.

When an Alt-Svc response header is received from an origin, its value invalidates and replaces all cached alternate services for that origin.

See Section 3.1.2 for general requirements on caching alternate services.

Note that the freshness lifetime for HTTP caching (here, 600 seconds) does not affect caching of Alt-Svc values.

3.3. Proposal: ALTSVC Frame

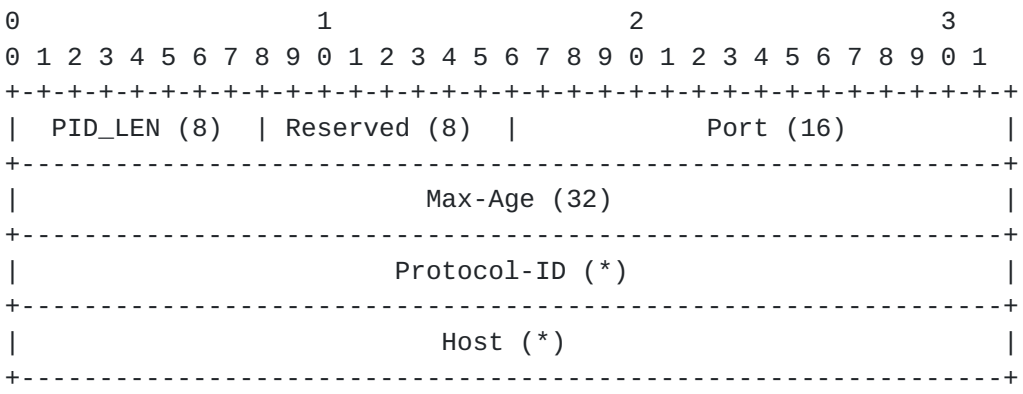
NOTE: Because of the current approach to HTTP/2 extensibility, this section will need to be incorporated to the frame type listing in HTTP/2 if accepted.

The ALTSVC frame (type=0xa) advertises the availability of an alternate service to the client. It can be sent at any time for an existing client-initiated stream or stream 0, and is intended to allow servers to load balance or otherwise segment traffic; see Section 3.1 for details.

An ALTSVC frame on a client-initiated stream indicates that the conveyed alternate service is associated with the origin of that stream.

An ALTSVC frame on stream 0 indicates that the conveyed alternate service is associated with all origins that map to the server's address (i.e., host and port).

The ALTSVC frame is intended for receipt by clients; a server that receives an ALTSVC frame MUST treat it as a connection error of type PROTOCOL_ERROR.



The ALTSVC frame contains the following fields:

- PID_LEN: An unsigned, 8-bit integer indicating the length, in bytes, of the PROTOCOL-ID field.
- Reserved: for future use.
- Port: An unsigned, 16-bit integer indicating the port that the alternate service is available upon.
- Max-Age: An unsigned, 32-bit integer indicating the freshness lifetime of the alternate service association, as per Section 3.1.2.
- Protocol-ID: A sequence of bytes (length determined by PID_LEN) containing the ALPN protocol identifier of the alternate service.
- Host: A sequence of bytes containing an ascii string indicating the host that the alternate service is available upon.

The ALTSVC frame does not define any flags.

3.4. Proposal: SETTINGS_UNIVERSAL_SCHEMES (4)

NOTE: This is a proposal for a new SETTINGS value, to be incorporated in that list if accepted.

A non-zero value indicates the sender MAY accept requests with schemes that do not match the default port for the server. A non-zero value is a pre-requisite for sending http:// over TLS via the previous https connection method described in Section 3.6.

3.5. Proposal: NOT_AUTHORITATIVE (13)

NOTE: This is a proposal for a new error code, which should be incorporated into the appropriate section if accepted.

The endpoint refuses the stream prior to performing any application processing. This server is not configured to provide a definitive response for the requested resource. Clients receiving this error code SHOULD remove the endpoint from their cache of alternate

services, if present.

3.6. Proposal: Discovery of TLS Support for http:// URIs

NOTE: This section, if accepted, ought to be added as a new subsection of "Starting HTTP/2".

A server wishing to advertise support for HTTP/2 over TLS for http:// URIs MAY do so by including an Alt-Svc (Section 3.2 response header with the "h2t" protocol identifier.

For example, a HTTP/1 connection could indicate support for HTTP/2 on port 443 for use with future http:// URI requests with this Alt-Svc header:

```
HTTP/1.1 200 OK
Alt-Svc: "h2t"=443
```

The process for starting HTTP/2 over TLS for an http:// URI is the same as the connection process for an https:// URI, except that authentication of the TLS channel is not required. The client MAY use either anonymous or authenticated ciphersuites. If an authenticated ciphersuite is used, the client MAY ignore authentication failures. This enables servers that only serve http:// URIs to use credentials that are not tied to a global PKI, such as self-signed certificates. Clients MAY reserve the use of certain security sensitive optimizations, such as caching the existence of this successful connection, for authenticated connections.

Additionally, if a client has previously successfully connected to a given server over TLS, for example as part of an https:// request, then it MAY attempt to use TLS for requests certain http:// URIs. To use this mechanism the server MUST have sent a non-zero SETTINGS_UNIVERSAL_SCHEMES setting to indicate support for this mechanism.

Eligible http:// URIs:

1. Contain the same host name as the URI accessed over TLS, and
2. Do not contain an explicit port number.

For example, if the client has successfully made a request for the URI "https://example.com/foo", then it may attempt to use TLS to make a request for the URI "http://example.com/bar", but not for the URI "http://example.com:80/". In particular, if a client has a TLS connection open to a server (for example, due to a past "https" request), then it may re-use that connection for "http" requests,

subject to the constraints above.

4. Security Considerations

Identified security considerations should be enumerated in the appropriate documents depending on which proposals are accepted. Those listed below are generic to all uses of alternate services; more specific ones might be necessary.

4.1. Changing Ports

Using an alternate service implies accessing an origin's resources on an alternate port, at a minimum. An attacker that can inject alternate services and listen at the advertised port is therefore able to hijack an origin.

For example, an attacker that can add HTTP response header fields can redirect traffic to a different port on the same host using the Alt-Svc header field; if that port is under the attacker's control, they can thus masquerade as the HTTP server.

This risk can be mitigated by restricting the ability to advertise alternate services, and restricting who can open a port for listening on that host.

4.2. Changing Hosts

When the host is changed due to the use of an alternate service, it presents an opportunity for attackers to hijack communication to an origin.

For example, if an attacker can convince a user agent to send all traffic for "innocent.example.org" to "evil.example.com" by successfully associating it as an alternate service, they can masquerade as that origin. This can be done locally (see mitigations above) or remotely (e.g., by an intermediary as a man-in-the-middle attack).

This is the reason for the requirement in Section 3.1.1 that any alternate service with a host different to the origin's be strongly authenticated with the origin's identity; i.e., presenting a certificate for the origin proves that the alternate service is authorized to serve traffic for the origin.

However, this authorization is only as strong as the method used to authenticate the alternate service. In particular, there are well-known exploits to make an attacker's certificate appear as

legitimate.

Alternate services could be used to persist such an attack; for example, an intermediary could man-in-the-middle TLS-protected communication to a target, and then direct all traffic to an alternate service with a large freshness lifetime, so that the user agent still directs traffic to the attacker even when not using the intermediary.

As a result, there is a requirement in Section 3.1.2 to examine cached alternate services when a network change is detected.

4.3. Changing Protocols

When the ALPN protocol is changed due to the use of an alternate service, the security properties of the new connection to the origin can be different from that of the "normal" connection to the origin, because the protocol identifier itself implies this.

For example, if a "https://" URI had a protocol advertised that does not use some form of end-to-end encryption (most likely, TLS), it violates the expectations for security that the URI scheme implies.

Therefore, clients cannot blindly use alternate services, but instead evaluate the option(s) presented to assure that security requirements and expectations (of specifications, implementations and end users) are met.

5. References

5.1. Normative References

[I-D.ietf-httpbis-p1-messaging]
Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", draft-ietf-httpbis-p1-messaging-26 (work in progress), February 2014.

[I-D.ietf-httpbis-p6-cache]
Fielding, R., Nottingham, M., and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Caching", draft-ietf-httpbis-p6-cache-26 (work in progress), February 2014.

[I-D.ietf-tls-applayerprotoneg]
Friedl, S., Popov, A., Langley, A., and S. Emile, "Transport Layer Security (TLS) Application Layer Protocol

Negotiation Extension", draft-ietf-tls-applayerprotoneg-04 (work in progress), January 2014.

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, December 2011.

5.2. Informative References

[I-D.ietf-httpbis-http2]
Belshe, M., Peon, R., Thomson, M., and A. Melnikov, "Hypertext Transfer Protocol version 2.0", draft-ietf-httpbis-http2-09 (work in progress), December 2013.

[I-D.nottingham-http2-encryption]
Nottingham, M., "Opportunistic Encryption for HTTP URIs", draft-nottingham-http2-encryption-02 (work in progress), December 2013.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

Appendix A. Acknowledgements

Thanks to Eliot Lear, Stephen Farrell, Guy Podjarny, Stephen Ludin, Erik Nygren, Paul Hoffman, Adam Langley, Will Chan and Richard Barnes for their feedback and suggestions.

The Alt-Svc header field was influenced by the design of the Alternate-Protocol header in SPDY.

Appendix B. TODO

- DNS: Alternate services are also amenable to DNS-based discovery. If there is sufficient interest, a future revision may include a proposal for that.
- Indicating Chosen Service: It's likely necessary for the server to know which protocol the user agent has chosen, and perhaps even the hostname (for load balancing). This could be conveyed as part of the "magic", or as a request header.
- Advice for setting headers: guidelines for servers that use the Alt-Svc header field.
- For the load balancing use case, it's necessary for clients to always flush altsvc cache upon a network change, but right now they're only required to examine the cache for suspicious entries. We should discuss whether this should be upgraded to always flush.

Authors' Addresses

Mark Nottingham
Akamai

Email: mnot@mnot.net
URI: http://www.mnot.net/

Patrick McManus
Mozilla

Email: mcmanus@ducksong.com
URI: https://mozillians.org/u/pmcmanus/