Hypertext Transfer Protocol Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 27, 2016

C. Benfield
July 26, 2015

Peer-to-peer Extension to HTTP/2
draft-benfield-http2-p2p-01

Abstract

This document introduces a negotiated extension to HTTP/2 that turns a single HTTP/2 connection into a bi-directional communication channel.

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 27, 2016.

Copyright Notice

Table of Contents

1. Introduction

The HTTP/2 [RFC7540] specification provides an alternative framing layer for the semantics of HTTP/1.1 [RFC7231]. This framing layer in principle allows for both parties in a HTTP/2 session to send requests and responses. However, the HTTP/2 specification also requires that the semantics of HTTP/1.1 be preserved. This means that one party of the conversation is considered the client, and one the server. Only the client may send requests, and only the server may send responses.

This document introduces an extension that can be advertised by a HTTP/2 client. This extension allows both the client and the server to send requests and responses. Essentially, this extension changes the protocol such that the notion of 'client' and 'server' are defined on a per-stream basis, rather than a per-connection basis.

The principle of this extension is similar to the Reverse HTTP [I-D.lentczner-rhttp] proposal made in 2009. HTTP/2's framing makes this a substantially more flexible extension than Reverse HTTP by allowing the client and server to vary on a per-stream basis, rather than affecting the whole connection.

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 RFC 2119 [RFC2119].

2. Additions to HTTP/2

This document introduces a new HTTP/2 setting ([RFC7540], Section 11.3) and a new HTTP/2 frame type ([RFC7540], Section 11.2), to allow for a HTTP/2 client to advertise its support for receiving server-initiated streams, and to allow a server to advertise its support for receiving client-initiated pushed streams.

The setting, SETTINGS_PEER_TO_PEER, is a negotiated setting ([RFC7540], Section 5.5).

2.1. SETTINGS_PEER_TO_PEER Setting

The following new SETTINGS parameters ([RFC7540], Section 6.5.2) are defined:

- SETTINGS_PEER_TO_PEER (0xTBA): Informs the remote endpoint of whether the sender supports the peer-to-peer extension to HTTP/2. A value of 1 indicates that the peer-to-peer extension is supported. Any other value, or the absence of this setting, indicates that the peer-to-peer extension is not supported.

  This setting MUST NOT be emitted by the server on the HTTP/2 connection. If the client receives this setting from the server it MUST respond with a connection error ([RFC7540] Section 5.4.1) of type PROTOCOL_ERROR.

2.2. CLIENT_AUTHORITY Frame

This document introduces the CLIENT_AUTHORITY frame. This frame MUST be emitted by a client after it sends a value of SETTINGS_PEER_TO_PEER of 1, and MAY be emitted by a client any time after. The purpose of this frame is to allow a client to advertise the authority or authorities for which it is prepared to accept requests.

This frame always applies to a whole connection. Therefore, the stream identifier for CLIENT_AUTHORITY frames MUST be 0. If a server receives a CLIENT_AUTHORITY frame whose stream identifier field is anything other than 0, it MUST respond with a connection error ([RFC7540] Section 5.4.1) of type PROTOCOL_ERROR.

2.2.1. Payload

Each CLIENT_AUTHORITY frame is made up of one or more of the following authority segments:

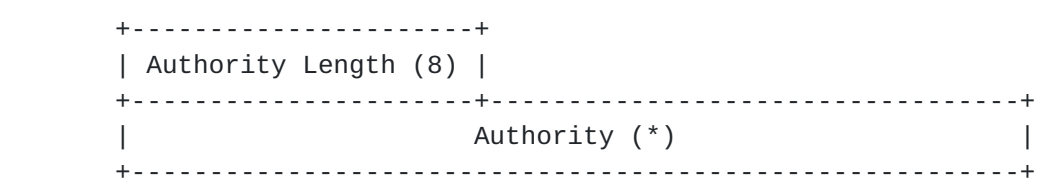

Figure 1: Client Authority Frame Payload

Each segment begins with a one-byte field indicating the length of the authority string the client is asserting. That field is then followed by a single authority field. The authority MUST be sent in whatever character encoding is going to be expected by the client on receipt of the :authority pseudo-header field.

2.2.2. Semantics

Generally speaking, a server or coalescing intermediary has no in-band method of validating that a client's authority claims are valid. Therefore, a conforming server MUST confirm a client's authority claims using some out-of-band method: see Section 3 for more.

A client MAY send a CLIENT_AUTHORITY frame at any time after the HTTP/2 preamble is complete. Each CLIENT_AUTHORITY frame is considered to be a complete list of authorities: therefore, a server MUST disregard all prior CLIENT_AUTHORITY frames when a new one is received. Also, servers MUST validate the asserted authorities for all CLIENT_AUTHORITY frames, not just the first one.

2.3. HTTP Changes

From the perspective of other HTTP RFCs, such as RFC 7231 [RFC7231] and RFC 7540 [RFC7540], this extension changes whether a peer is considered a 'client' or a 'server' on a per-stream basis, instead of a per-connection basis, based on which peer opened the stream and how they did so. If a stream is initiated by a HEADERS frame, the peer that sent the HEADERS frame is considered the 'client' for the remainder of the lifetime of that stream, while the other peer is considered the 'server'.

Otherwise, the new definition of 'client' and 'server' is preserved for the purposes of the PUSH_PROMISE frame ([RFC7540], Section 6.6). As a result, whichever peer is considered the 'server' for a given stream can push other streams to the 'client' peer.

The rest of the requirements of RFC 7231 [RFC7231] are preserved.

2.4. Client Behavioral Changes

When a client emits the SETTINGS_PEER_TO_PEER setting with a value of 1, it is informing the server that it is willing to accept HTTP requests from the server, allowing the server to open streams with HEADERS frames. This lifts some of the restrictions of RFC 7540 [RFC7540] Section 8.

If a client has sent the SETTINGS_PEER_TO_PEER setting with a value of 1, the client MUST NOT reject an attempt by the server to change the value of SETTINGS_ENABLE_PUSH to 1.

If the client, subsequent to sending SETTINGS_PEER_TO_PEER with value 1, receives from the server a value of SETTINGS_ENABLE_PUSH of 1, it MAY open streams by sending PUSH_PROMISE frames. The client MUST NOT send a PUSH_PROMISE frame on a stream that it opened by means of a HEADERS frame: only server-initiated streams may be used for sending PUSH_PROMISE frames. All other limitations about PUSH_PROMISE frames in RFC 7540 [RFC7540] continue to apply, except that the words 'server' and 'client' are defined on a per-stream basis.

2.5. Server Behavioral Changes

When a server receives the SETTINGS_PEER_TO_PEER setting from the client with a value of 1, it MAY at any point afterwards issue a non-zero value for SETTINGS_ENABLE_PUSH. This allows clients to open streams with PUSH_PROMISE and also lifts some of the restrictions of RFC 7540 [RFC7540] Section 8: specifically those sections that only allow servers to send PUSH_PROMISE frames, and only allow clients to receive them.

If the client attempts to send a PUSH_PROMISE frame on a stream that was opened by the client (by sending a HEADERS frame), the server MUST treat this event as a connection error ([RFC7540] Section 5.4.1) of type PROTOCOL_ERROR.

2.6. Other Extensions

When this extension is deployed with other extensions to HTTP/2, the behaviour of this extension does not change. All other extensions that refer to 'client' or 'server' SHOULD be treated as though those terms apply on a per-stream basis.

If other extensions apply 'server' or 'client' to the whole connection (e.g. for settings in SETTINGS frames, which are sent on

stream 0), then both peers SHOULD be considered clients and both peers should be considered servers.

3. Authority Validation

Generally speaking, a server or coalescing intermediary has no in-band method of validating that a client's authority claims are valid. Therefore, a conforming server MUST confirm a client's authority claims using some out-of-band method.

This specification does not lay out in detail any proposed mechanism for doing this validation, as the best approach may vary from deployment to deployment. However, some options include:

- validating authorities against a TLS certificate presented by the client during TLS handshake.
- confirming that a reverse DNS lookup for the client IP returns the authority asserted by the client.
- a static list of IP addresses trusted for a given authority.

The only requirement is that a server MUST implement some form of validation, and then MUST treat any attempt by a client to assert an authority that it cannot validate as a connection error ([RFC7540] Section 5.4.1) of type PROTOCOL_ERROR.

4. IANA Considerations

4.1. HTTP/2 Frame Type Registry Update

This document updates the HTTP/2 Frame Type registry ([RFC7540], Section 11.2). The entries in the following table are registered by this document.

Name	Code	Section
CLIENT_AUTHORITY	TBD	Section 2.2

4.2. HTTP/2 Settings Registry Update

This document updates the registry for HTTP/2 Settings ([RFC7540], Section 11.4). The entries in the following table are registered by this document.

Name	Code	Initial Value	Section
PEER_TO_PEER	TBD	0	Section 2.1

5. Acknowledgements

Thanks to Fedor Indutny for the original idea, and Amos Jeffries, Mike Bishop, and Ilari Liusvaara for their follow-up.

Thanks also to Tyrel Souza, Donald Stufft, and Paul Kehrer for proofreading.

Thanks to David Reid for pointing out the Reverse HTTP proposal [I-D.lentczner-rhttp].

Thanks to Amos Jeffries for proposing an advertised extension, rather than a negotiated one.

6. References

6.1. 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>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

[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>.

6.2. Informative References

[I-D.lentczner-rhttp] Lentczner, M. and D. Preston, "Reverse HTTP", draft-lentczner-rhttp-00 (work in progress), March 2009.

Appendix A. Changelog

(This appendix to be deleted by the RFC Editor.)

Since -00:

- Clarified the semantics behind multiple CLIENT_AUTHORITY frames.
- Removed the requirement for servers to issue SETTINGS_PEER_TO_PEER, instead allowing the extension to be purely client-advertised.

Author's Address

Cory Benfield

Email: cory@lukasa.co.uk