httpbis Working Group
Internet-Draft
Intended status: Standards Track
Expires: 6 October 2019

G. Xie
F. Frindell
Facebook Inc.
4 April 2019

An HTTP/2 extension for bidirectional messaging communication
draft-xie-bidirectional-messaging-01

Abstract

This draft proposes a http2 protocol extension, which enables bidirectional messaging communication between client and server.

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 6 October 2019.

Copyright Notice

Copyright (c) 2019 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

HTTP/2 is the de facto application protocol in Internet. The optimizations developed in HTTP/2, like stream multiplexing, header compression, and binary message framing are very generic. They can be useful in non web browsing applications, for example, Publish/ Subscribe, RPC. However, the request/response from client to server communication pattern limits HTTP/2 from wider use in these applications. This draft proposes a HTTP/2 protocol extension, which enables bidirectional messaging between client and server.

The only mechanism HTTP/2 provides for server to client communication is PUSH_PROMISE. While this satisfies some use-cases, it is unidirectional, i.g. the client cannot respond. In this draft, a new frame is introduced which has the routing properties of PUSH_PROMISE and the bi-directionality of HEADERS. Further, clients are also able group streams together for routing purposes, such that each individual stream does not need to carry additional routing information.

2. Conventions and Terminology

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC2119].

All the terms defined in the Conventions and Terminology section in [RFC7540] apply to this document.

3. Solution Overview

3.1. Routing Stream and ExStream

A routing stream (RStream) is a long lived HTTP/2 stream in nature. RStreams are initiated by clients, and can be routed independently by any intermediaries. Though an RStream is effectively a regular HTTP/2 stream, RStreams are recommended for exchanging metadata, but not user data.

A new HTTP/2 stream called ExStream is introduced for exchanging user data. ExStreams are recommended for short lived transactions, so intermediaries and servers can gracefully shutdown ExStreams within a short time. The typical use case can be a subscription or publish request/response in Publish/Subscribe use case, or an RPC call between two endpoints.

An ExStream is opened by an EX_HEADERS frame, and continued by CONTINUATION and DATA frames. An ExStream MUST be associated with an open RStream, and MUST NOT be associated with any other ExStream. ExStreams are routed according to their RStreams by intermediaries and servers. Effectively, all ExStreams with the same RStream form a logical stream group, and are routed to the same endpoint.

3.2. Bidirectional Messaging Communication

With RStreams and ExStreams, HTTP/2 can be used for bidirectional messaging communication. As shown in the follow diagrams, after an RStream is open from client to server, either endpoint can initiate an ExStreams to its peer.

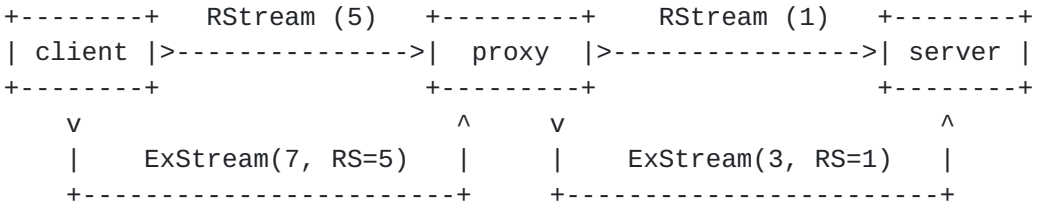

Figure 1: Client initiates the ExStream to server, after an RStream is open.

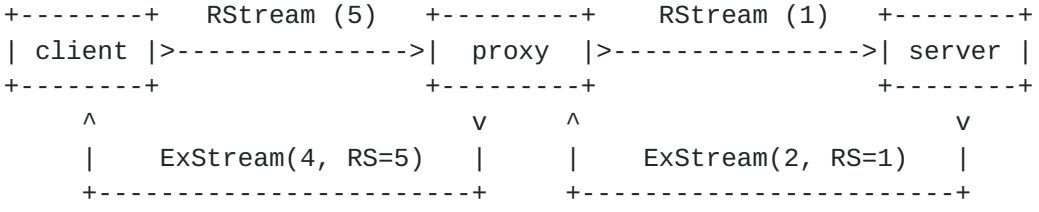

Figure 2: Server initiates the ExStream to client, after an RStream is open.

Beyond that, clients can multiplex RStreams, ExStreams and regular HTTP/2 streams into one HTTP/2 connection. This enables clients to access different services without initiating new TCP connections. This avoids the latency cost of setting up new connections. This is more desirable for mobile devices because they usually have longer RTT and battery constraints. Multiplexing these services also allows them to share a single TCP connection congestion control context.

As shown in the following diagram, the client can exchange data with PubSub, RPC and CDN three different services with one TCP connection.

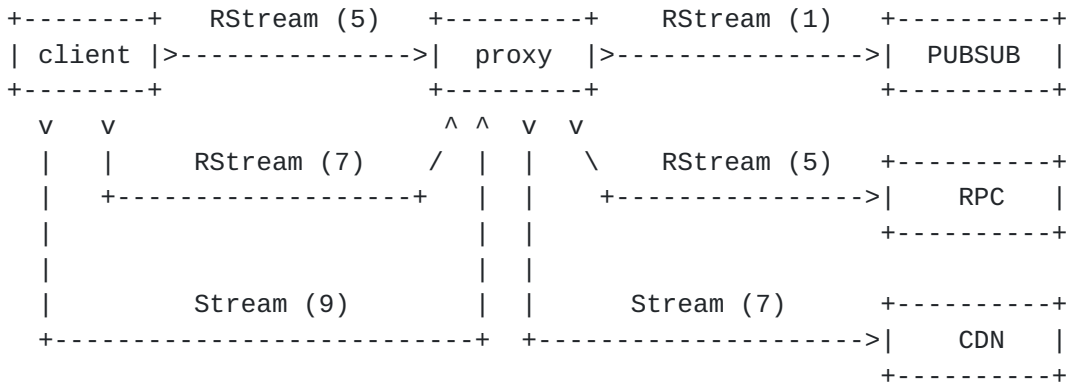

Figure 3: Client opens multiple RStreams and a HTTP/2 stream within one HTTP/2 connection.

3.3. States of RStream and ExStream

RStreams are regular HTTP/2 streams that follow the stream lifecycle in [RFC7540], section 5.1. ExStreams use the same lifecycle as regular HTTP/2 streams, but also depend on their RStreams. If a RStream is reset, endpoints MUST reset the ExStreams associated with that RStream. If the RStream is closed, endpoints SHOULD allow existing ExStreams complete normally. The RStream SHOULD remain open while communication is ongoing. Endpoints SHOULD refresh any timeouts on the RStream while associated ExStreams are open.

A sender MUST NOT initiate new ExStreams if on an RStream that is in the open or half closed (remote) state.

Endpoints process new ExStreams only when the RStream is open or half closed (local) state. If an endpoint receives an EX_HEADERS frame specifying an RStream in the closed or haf closed (remote) state, it MUST respond with a connection error of type ROUTING_STREAM_ERROR.

3.4. Negotiate the Extension through SETTINGS frame

The extension SHOULD be disabled by default. Endpoints can negotiate the use of the extension through the SETTINGS frame. If an implementation supports the extension, it is RECOMMENDED to include the ENABLE_EX_HEADERS setting in the initial SETTINGS frame. HTTP/2 compliant implementations will ignore the setting if it is unknown. An endpoint can send EX_HEADERS frames immediately upon receiving a SETTINGS frame with ENABLE_EX_HEADERS=1.

Endpoints MUST NOT send out EX_HEADERS before receiving a SETTINGS frame with the ENABLE_EX_HEADERS=1. If a remote endpoint does not support this extension, the EX_HEADERS will be ignored, making the header compression contexts inconsistent between sender and receiver.

If an endpoint supports this extension, but receives EX_HEADERS before ENABLE_EX_HEADERS, it MUST respond with a connection error EX_HEADER_NOT_ENABLED_ERROR.

Intermediaries SHOULD send the ENABLE_EX_HEADERS setting to clients, only if intermediaries and their upstream servers can support this extension. If an intermediary receives an ExStream but discovers the destination endpoint does not support the extension, it MUST reset the stream with EX_HEADER_NOT_ENABLED_ERROR.

3.5. Interaction with standard http2 features

The extension implementation should apply stream and connection level flow control, maximum concurrent streams limit, GOAWAY logic to both RStreams and ExStreams.

4. HTTP2 EX_HEADERS Frame

The EX_HEADERS frame (type=0xfb) has all the fields and frame header flags defined by HEADERS frame in HEADERS [RFC7540]. Moreover, EX_HEADERS has one extra field, RStream ID. It is used to open an ExStream, and additionally carries a header block fragment. EX_HEADERS frames can be sent on a stream in the "idle", "open", or "half-closed (remote)" state.

Like HEADERS, the CONTINUATION frame (type=0x9) is used to continue a sequence of header block fragments, if the headers do not fit into one EX_HEADERS frame.

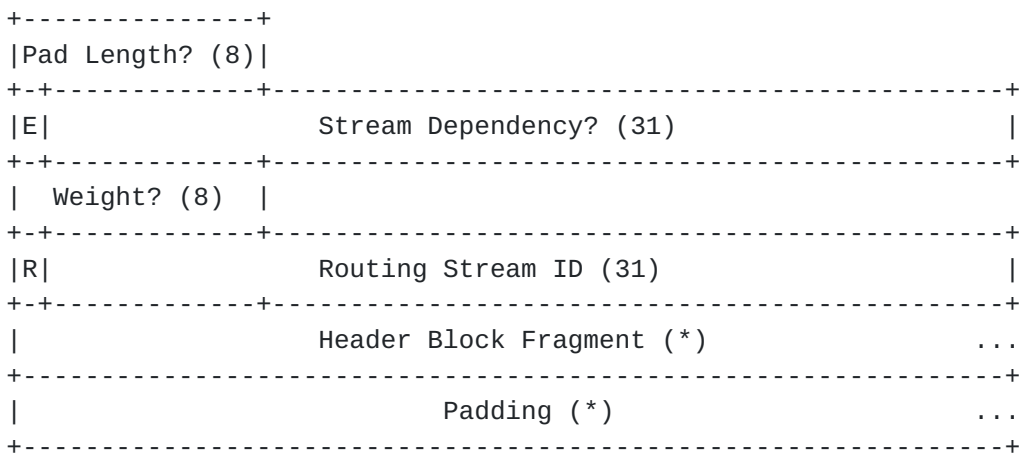

Figure 4: EX_HEADERS Frame Payload

The RStream specified in EX_HEADERS frames MUST be an open stream. The recipient MUST respond with a connection error ROUTING_STREAM_ERROR PROTOCOL_ERROR, if the specified RStream is missing; or is an ExStream rather than a stream; or is closed or half-closed (remote). Otherwise, the states maintained for header compression or flow control) may be out of sync.

5. IANA Considerations

This document establishes a registry for a new frame type, setting, and error code.

5.1. FRAME TYPE Registry

The entry in the following table are registered by this document.

Frame Type	Code	Section
EX_HEADERS	0xfb	

5.2. Settings Registry

The entry in the following table are registered by this document.

Name	Code	Initial Value	Specification
ENABLE_EX_HEADERS	0xfbfb	0	

5.3. Error Code Registry

The entry in the following table are registered by this document.

Name	Code	Description	Specification
ROUTING_STREAM_ERROR	0xfb	Routing stream is not open	
EX_HEADERS_NOT_ENABLED_ERROR	0xfc	EX_HEADERS is not enabled yet	

6. 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, <https://www.rfc-editor.org/info/rfc2119>.

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

Authors' Addresses

Guowu Xie
Facebook Inc.
1 Hacker Way
Menlo Park

Email: woo@fb.com

Alan Frindell
Facebook Inc.

Email: afrind@fb.com