WEBTRANS BOF
IETF 106

Singapore

Wednesday, November 20, 2019

13:30 – 15:00

Room: Collyer

Meetecho: https://www.meetecho.com/ietf106/webtrans/
Mailing list: webtransport@ietf.org
Jabber Room: webtrans@jabber.ietf.org

Note Well

This is a reminder of IETF policies in effect on various topics such as patents or code of conduct. It is only meant to point you in the right direction. Exceptions may apply. The IETF's patent policy and the definition of an IETF "contribution" and "participation" are set forth in BCP 79; please read it carefully.

As a reminder:

- By participating in the IETF, you agree to follow IETF processes and policies.
- If you are aware that any IETF contribution is covered by patents or patent applications that are owned or controlled by you or your sponsor, you must disclose that fact, or not participate in the discussion.
- As a participant in or attendee to any IETF activity you acknowledge that written, audio, video and photographic records of meetings may be made public.
- Personal information that you provide to IETF will be handled in accordance with the IETF Privacy Statement.
- As a participant or attendee, you agree to work respectfully with other participants; please contact the ombudsteam (https://www.ietf.org/contact/ombudsteam/) if you have questions or concerns about this.

Definitive information is in the documents listed below and other IETF BCPs. For advice, please contact WG Chairs or ADs.

- BCP 9 (Internet Standards Process)
- BCP 25 (Working Group processes)
- BCP 25 (Anti-Harassment Procedures)
- BCP 54 (Code of Conduct)
- BCP 78 (Copyright)
- BCP 79 (Patents, Participation)
- https://www.ietf.org/privacy-policy/ (Privacy Policy)

Agenda

- 13:30 – 13:40 Preliminaries, Chairs (10 minutes)
  - Note Well, Blue Sheets, Note Takers, and Jabber Scribe
  - BoF Context and Agenda Bash
- 13:40 – 13:50 WebTransport Overview and Requirements, Victor Vasiliev (10 minutes)
  - https://tools.ietf.org/html/draft-vvv-webtransport-overview
- 13:50 – 14:25 Relevant Drafts (35 minutes)
  - An Unreliable Datagram Extension to QUIC, Eric Kinnear & Tommy Pauly (5 minutes)
    - https://tools.ietf.org/html/draft-pauly-quic-datagram, draft-schinazi-quic-h3-datagram
  - HTTP/2 as a Transport for Arbitrary Bytestreams, Eric Kinnear & Tommy Pauly (5 minutes)
    - https://tools.ietf.org/html/draft-kinnear-httpbis-http2-transport
  - An HTTP/2 Extension for Bidirectional Message Communication, Guowu Xie & Alan Frindell (10 minutes)
    - https://tools.ietf.org/html/draft-xie-bidirectional-messaging
  - WebTransport over QUIC, Victor Vasiliev (5 minutes)
    - https://tools.ietf.org/html/draft-vvv-webtransport-quic
  - WebTransport over HTTP/3, Victor Vasiliev (10 minutes)
    - https://tools.ietf.org/html/draft-vvv-webtransport-http3
- 14:25 – 14:50 Q&A (25 minutes)
- 14:50 – 14:55 Pointer to Charter Discussion, Chairs (5 minutes)
- 14:55 – 15:00 Wrap up and Summary, Chairs & ADs (5 minutes)

Time Management

To ensure that we have adequate time for general questions, we will be enforcing strict time limits during the presentations.

After each presentation, if there is time remaining, we will open the mic for clarifying questions only.

After all the presentations, there is a Q&A agenda item where all questions will be welcome.

Please relinquish the mic when time runs out.

BOF Context

- This is a non-WG forming BOF.
- The focus is on client/server protocols (not APIs).
- We assume familiarity with the following RFCs:
  - RFC 6455: The Websocket Protocol
    - Uses the HTTP 1.1 Upgrade mechanism to transition a TCP connection from HTTP to a Websocket Connection.
  - RFC 8441: Bootstrapping Websockets over HTTP/2
    - Extends the HTTP CONNECT method as specified for HTTP/2 (RFC 7540).
    - Provides a tunnel on a single HTTP/2 stream that can carry data (and is multiplexed with other streams).

Central Question: “What’s Next?”

- We will hear a proposal for datagram transport (QUIC and HTTP/3).
  - This material is only provided for background, since it is likely to be handled in (and was just presented to) the QUIC WG.
- We will also hear proposals in two categories:
  - Proposals for extending the HTTP CONNECT method for HTTP/2.
  - Proposal(s) for the WebTransport Protocol Framework, which includes:
    - Support for uni and bi-directional reliable streams
    - Unreliable transport of datagrams in either direction
    - Potential operation over HTTP/3 and QUIC
    - Fallback considerations (e.g. if HTTP/3 and QUIC are not available)

WebTransport Overview and Requirements (10 minutes)

Presentation End: 13:50

Victor Vasiliev

https://tools.ietf.org/html/draft-vvv-webtransport-overview

Bidirectional Communication on the Web

<table>
<tr><th></th><th>Client-Server</th><th>Peer-to-peer</th></tr>
<tr><td>Reliable and ordered</td><td>WebSocket</td><td rowspan="3">RTCDataChannel</td></tr>
<tr><td>Reliable but unordered</td><td rowspan="2">?</td></tr>
<tr><td>Unreliable and unordered</td></tr>
</table>

Bidirectional Communication on the Web (proposed)

<table>
<tr><th></th><th>Client-Server</th><th>Peer-to-peer</th></tr>
<tr><td>Reliable and ordered</td><td>WebSocket
(also WebTransport!)</td><td rowspan="3">RTCDataChannel</td></tr>
<tr><td>Reliable but unordered</td><td rowspan="2">WebTransport</td></tr>
<tr><td>Unreliable and unordered</td></tr>
</table>

WebTransport features

- Streams
  - Arbitrary sized
  - Reliable
  - Independent (when possible)
  - Cancellable (when possible)
- Datagrams
  - MTU-sized
  - Unreliable (when possible)

WebTransport requirements

Required from any transport within WebTransport scope:

- TLS for confidentiality and authentication
- Congestion control
- Origin checks
- Prevent cross-protocol attacks
- Continuously maintain consent to send data
- Identifiable using a URI

Target applications

Anything that wants one of the following:

- "WebSockets for UDP"
- "WebSockets without head-of-line blocking"

We've reached out to a wide range of web developers, and there is plenty of interest in this in following domains:

- Machine learning
- Web games
- Live streaming
- Cloud gaming
- Remote desktop
- Web chat

Machine Learning Example

- Context: Machine Learning that cannot be handled on the device (e.g. client/server)
- 1st generation APIs: REST
  - Example: POST an image, receive a response describing regions and what was recognized within them.
- 2nd generation APIs: Websockets
  - Lower latency compared with REST
  - Example: Speech transcription (send speech, receive transcript)
- Goals for 3rd generation APIs
  - Even lower latency (e.g. removal of HOL blocking)
  - Ability to mix datagrams, uni-directional and bi-directional streams on the same connection.
  - Examples: speech translation, emotion analysis (audio or video)

Overview of proposed transports

	Dedicated	Pooled
QUIC-based	QuicTransport	Http3Transport
TCP-based (fallback)	FallbackTransport	Http2Transport

QuicTransport vs Http3Transport

QuicTransport:

- Dedicated connection
  - Transport fine-tuning on server
  - More stats exposed to client
- No HTTP/3 dependency
- Target applications:
  - Video games on the Web
  - Real-time media

Http3Transport:

- Pooled with other HTTP/3 traffic
- HTTP features:
  - Use HTTP load balancing and routing
  - Headers for metadata
- Target applications:
  - General web applications
  - Web chats
  - Push notifications

TCP-based fallback

What to do when QUIC is blocked?

- FallbackTransport: based on WebSocket
  - Polyfillable: can be used when browser does not support WebTransport at all
- Http2Transport: natural fallback for Http3Transport

An Unreliable Datagram Extension to QUIC (5 minutes)

Presentation End: 13:55

Tommy Pauly & Eric Kinnear

https://tools.ietf.org/html/draft-pauly-quic-datagram

https://tools.ietf.org/html/draft-schinazi-quic-h3-datagram

An Unreliable Datagram Extension to QUIC

draft-pauly-quic-datagram

Tommy Pauly (tpauly@apple.com)

Eric Kinnear (ekinnear@apple.com)

David Schinazi (dschinazi.ietf@gmail.com)

WEBTRANS

IETF 106, November 2019, Singapore

Motivation

Unreliable data transmission supports many use cases

- Applications that need a reliable control stream and unreliable flows
- Media streaming, gaming, VPN-style tunneling, and more

QUIC provides functionality beyond that of DTLS, UDP

Let's use the QUIC extension mechanism!

Why Datagrams in QUIC?

Share a single handshake and authentication context for reliable stream data and unreliable datagram data

QUIC handshake has more nuanced loss recovery during the handshake compared to DTLS

Use QUIC features not present in alternatives

- Transport parameters
- Transport level acknowledgements of datagram data
- Multiplexing of additional content over same transport

Design

`DATAGRAM` frame (`0x30` and `0x31`)

Length field is optional, determined by least significant bit

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        [Length (i)]                         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Datagram Data (*)                      ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Negotiated via `max_datagram_frame_size` transport parameter

Design Details

`DATAGRAM` frames are ack-eliciting and not retransmitted

- Just like `PING`

`DATAGRAM` frames do not contribute to flow control limits

Flow IDs are gone

- Didn’t go far, see draft-schinazi-quic-h3-datagram-02

`max_datagram_frame_size` can be stored for 0-RTT

Implementation Status

Supported by multiple implementations

- quiche (SiDUCK), aioquic, Google QUIC, AppleQUIC

Wireshark can dissect DATAGRAM frames

Achieved interop between quiche and aioquic during the hackathon

Questions?

DATAGRAM Frame

(`0x30` and `0x31`)

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         [Length (i)]                        ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Datagram Data (*)                      ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

HTTP/2 as a Transport for Arbitrary Bytestreams (10 minutes)

Presentation End: 14:00

Eric Kinnear & Tommy Pauly

https://tools.ietf.org/html/draft-kinnear-httpbis-http2-transport

Using HTTP/2 as a Transport for Arbitrary Bytestreams

draft-kinnear-httpbis-http2-transport

Eric Kinnear (ekinnear@apple.com)

Tommy Pauly (tpauly@apple.com)

WEBTRANS

IETF 106, November 2019, Singapore

Motivation

Generic transport for secure, multiplexed bytestreams

- These can be unidirectional or bidirectional
- Low setup cost for new streams
- Single congestion and recovery context

Peer-to-peer communication

- Example: Remote IPC

Share underlying transport with existing infrastructure

Why HTTP/2?

HTTP/2 provides framing layer with many desired transport features

- Configuration exchange
- Multiplexed streams
- Shared congestion control and loss recovery state
- Flow control
- Stream relationships and priorities
- Traverses the internet

Some of these properties are really coming from TLS/TCP

Potential Solution

CONNECT allows tunneling to another endpoint

Extended CONNECT allows connecting to server itself

HTTP headers enable additional negotiation

Coexists with standard HTTP request/response streams

Can also enable tunneling of UDP, with additional framing

New `:protocol` Values

Extended CONNECT defines `:protocol` value for use with WebSocket

Make generic by defining common base not specific to WebSocket

Define additional `:protocol` value

- “`bytestream`”
  - Direct stream mapping for arbitrary bytestreams to remote server

Individual applications can use specific `:protocol` values for negotiation

Motivation

Generic transport for secure, multiplexed bytestreams

- These can be unidirectional or bidirectional
- Low setup cost for new streams
- Single congestion and recovery context

Peer-to-peer communication

- Example: Remote IPC

Share underlying transport with existing infrastructure

Motivation

Generic transport for secure, multiplexed bytestreams

- These can be unidirectional or bidirectional
- Low setup cost for new streams
- Single congestion and recovery context

Peer-to-peer communication

- Example: Remote IPC, QUIC

Share underlying transport with existing infrastructure

Why QUIC?

HTTP/3 over QUIC falls back to HTTP/2 over TLS/TCP

What transport abstraction does QUIC alone use over TCP?

HTTP/2 provides framing layer with many desired transport features

- Configuration exchange
- Multiplexed streams
- Flow Control
- Stream relationships and priorities

TLS/TCP provides shared congestion control and loss recovery state

Solution

Extended CONNECT defines `:protocol` value for use with WebSocket

Make generic by defining common base not specific to WebSocket

Define additional `:protocol` value

- “`bytestream`”
  - Direct stream mapping for arbitrary bytestreams to remote server

Individual applications can use specific `:protocol` values for negotiation

Define new SETTING to allow bidirectional use of (Extended) CONNECT

Summary

Add new `:protocol` values to Extended CONNECT handshake

- Sharing multiple connections to server over single underlying transport
- Ability to proxy UDP traffic more effectively to (and through) the server
- Built in security with low setup cost for new streams

Add new SETTING to allow using Extended CONNECT in both directions

- Enables the benefits above for peer-to-peer communications
- Provides fallback mechanism for QUIC over HTTP/2 framing

Underlying Concepts

- Multiplex multiple protocols over a single transport connection
  - Method for negotiating use of stream for different protocol
  - Built in security with minimal setup cost for new streams
- Bidirectional establishment of streams
  - Must traverse intermediaries in both directions
- Can be extended to support unreliable delivery and datagrams

Questions?

An HTTP/2 Extension for Bidirectional Message Communication (10 minutes)

Presentation End: 14:10

Guowu Xie & Alan Frindell

https://tools.ietf.org/html/draft-xie-bidirectional-messaging

Extending HTTP/2 for Bidirectional Messaging

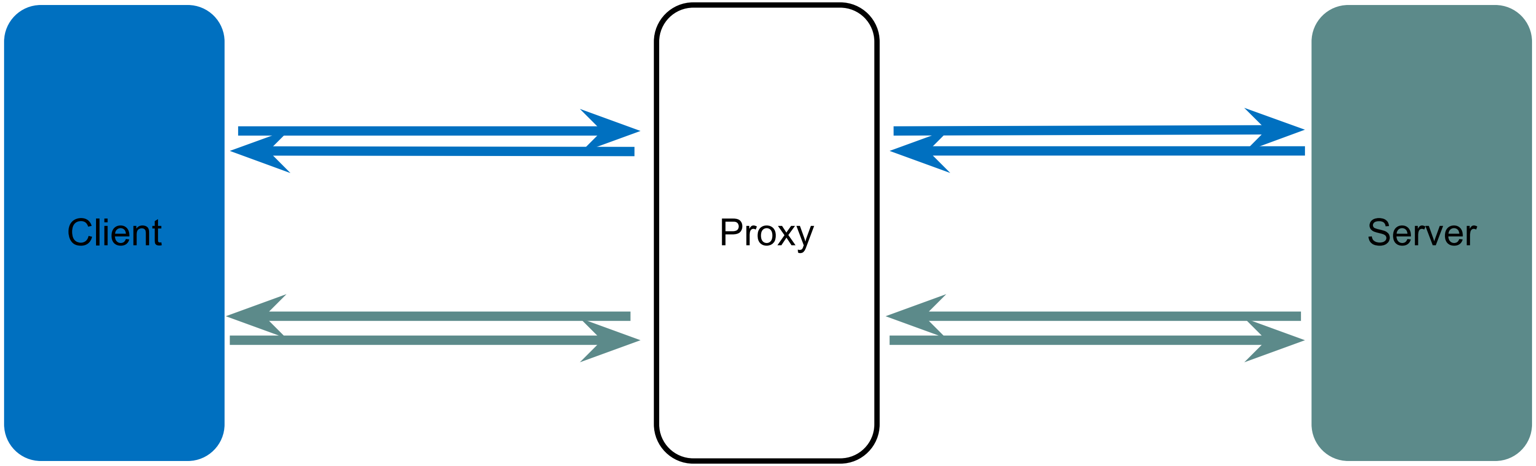
Guowu (Woo) Xie, Alan Frindell

{woo, afrind}@fb.com

WebTransport BOF

https://tools.ietf.org/html/draft-xie-bidirectional-messaging-02

Messaging requires bidirectional communication



Existing Solutions

- Stream Tunneling
- Server Push

Stream Tunneling

- Client establishes a tunnel to server over a single stream
  - Long polling
  - WebSocket
  - draft-kinnear-httpbis-http2-transport
- Server can initiate communication via this tunnel
- A HTTP/2 stream is treated like a socket
- Client and server have to speak some app protocol over the tunnel

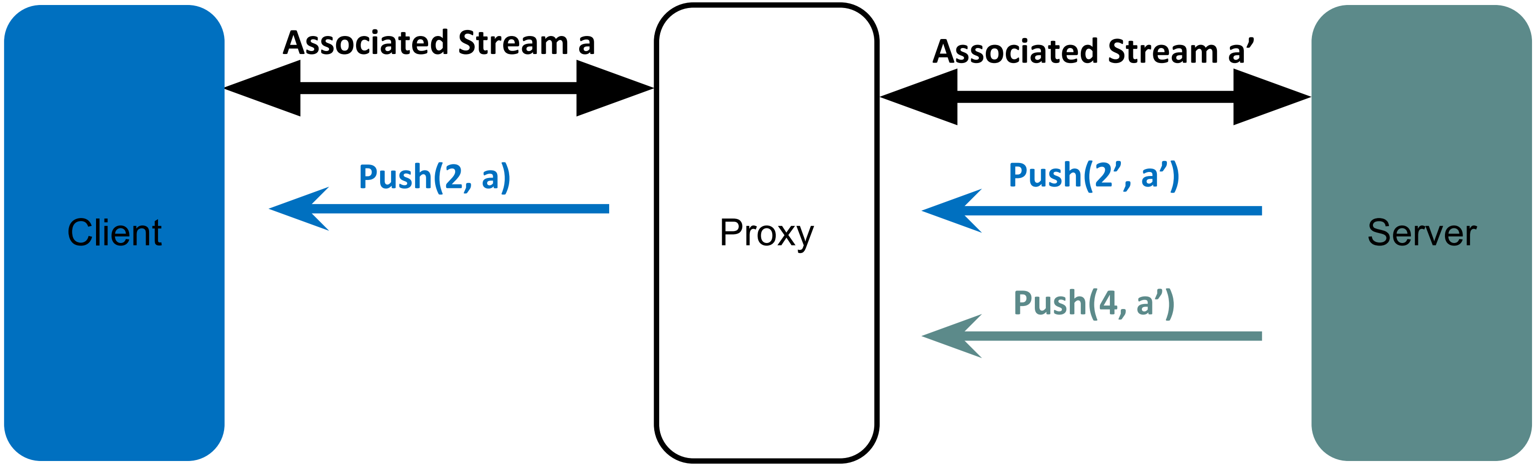
Limitations of Stream Tunneling

- Multiple layers of framing
  - HTTP/2 frame header + WebSocket header + app protocol header
- Forces web developers to design their own app protocols

- Reintroduces HoL blocking in HTTP/3
- Bypasses header compression
- Bypasses stream prioritization
- GOAWAY is less effective

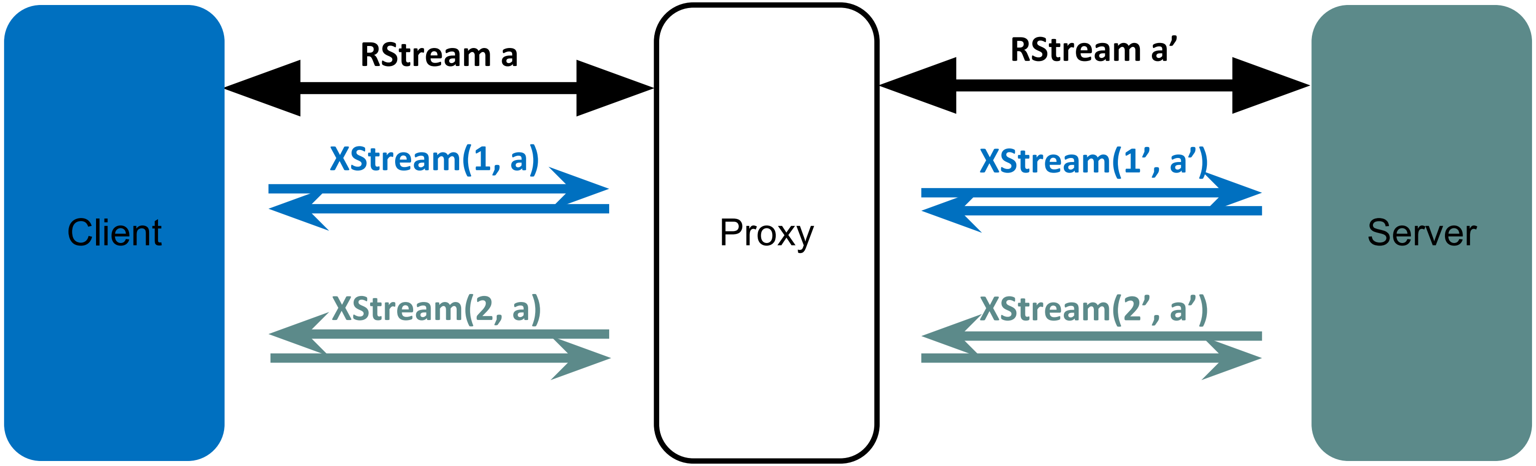
HTTP/2 Server Push?

- Unidirectional, server to client only
- Lack of acknowledgement makes it unsuitable for messaging

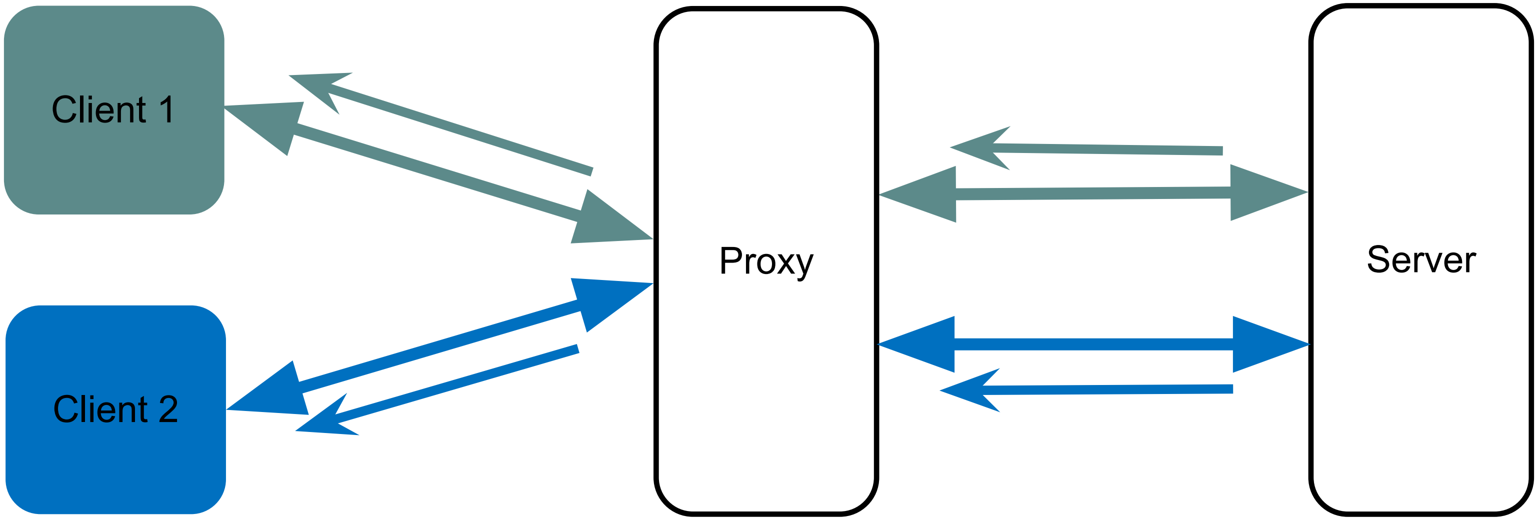


Extension Proposal

- Make “Associated Stream” generic: Routing Stream
- New Frame: EX_HEADERS
    - can be sent by either peer to open an eXtended Stream (XStream)
    - references an open Routing Stream

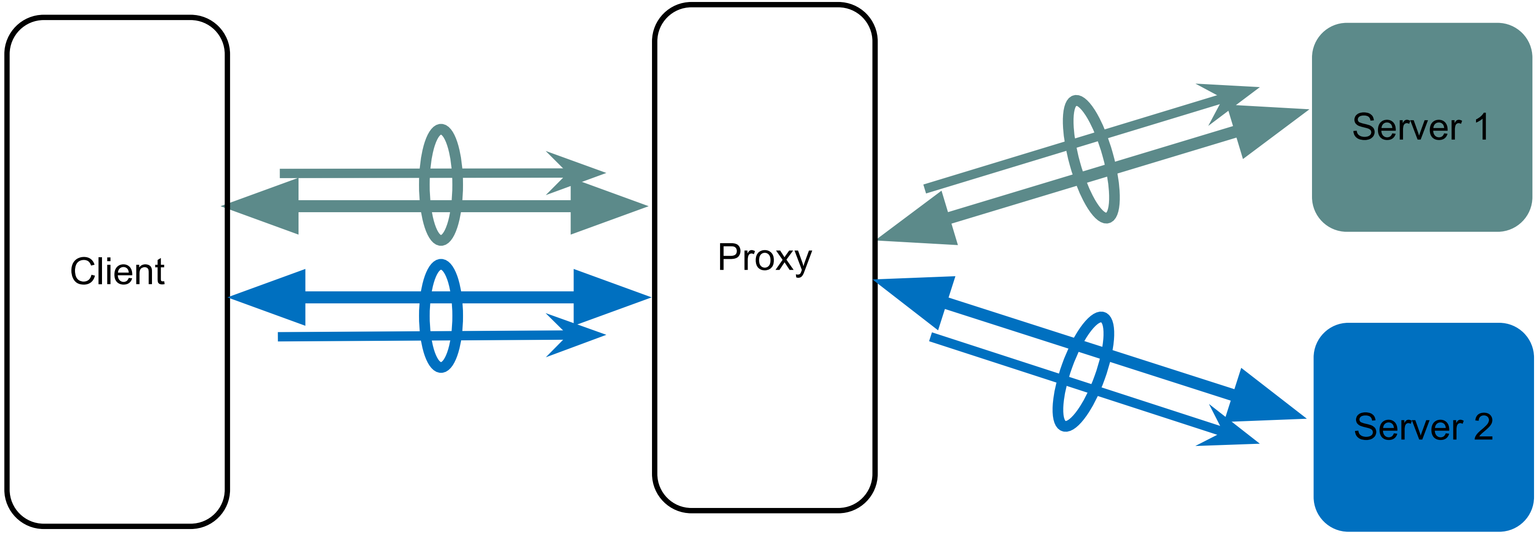


Intermediary Traversal



XStreams are routed via Routing Stream

Stream Grouping



Individual XStreams do not need to carry headers for routing

Comparison with WebTransport-over-h3

- Routing Stream vs WebTransport Session
    - Routing between server and client through intermediaries
    - Grouping dependant streams
- XStream vs WebTransport_stream
    - can be created by either peer
    - routing depends on Routing Streams or Session ID
- But...HTTP Message vs a stream of bytes
    - HTTP Message = structured meta-data (headers) + data (body)
    - better abstraction, and a richer building block

Q & A

details: https://tools.ietf.org/html/draft-xie-bidirectional-messaging-02

WebTransport over QUIC (5 minutes)

Presentation End: 14:15

Victor Vasiliev

https://tools.ietf.org/html/draft-schinazi-quic-h3-datagram

What is QuicTransport?

QuicTransport is an application protocol on top of QUIC.

- “WebSocket for QUIC”
- Design principle: minimal features added on top of QUIC
- Features required to meet WebTransport requirements:
  - ALPN value (“wq”)
  - Client indication (stream with origin)
  - URI scheme

Client indication

- Sent by the client on stream 2 (first client unidirectional stream)
- A set of key-value pairs
- Defined fields:
  - Origin (0x0000)
  - Path (0x0001)

QuicTransport URI scheme

```
quic-transport://server.test:50000/test?foo=bar
```

sent as SNI

sent in client indication

Example

1. A web app (on https://example.com) calls new `QuicTransport("quic-transport://server.test:50000/foo")`
2. Browser sends a QUIC ClientHello to `server.test` on port 50000 with ALPN list of "`wq`"
3. `server.test` receives it and sends a ServerHello with ALPN "`wq`"
4. Browser receives ServerHello and sends, on top of other QUIC packets, stream 2 with the following data and the FIN:
   - 0x0000 (origin): `https://example.com`
   - 0x0001 (path): `/foo`
5. Server receives the stream 2 and accepts the origin
6. The application can now send and receive streams and datagrams.

WebTransport over HTTP/3 (10 minutes)

Presentation End: 14:25

Victor Vasiliev

https://tools.ietf.org/html/draft-vvv-webtransport-http3

Http3Transport overview

Allows WebTransport sessions to happen within existing HTTP/3 connections

- A special transport parameter used to indicate support on both sides
- Extended CONNECT mechanism (RFC 8441) is used to create a session
- If the server accepts the session, it returns a new **session ID** in the response headers
- The session ID is used to associate all further streams and datagrams with the header
- All streams and datagrams have a special prefix indicating that the stream belongs to Http3Transport session and is not a regular HTTP stream

Http3Transport example

1. A web app (on https://example.com) calls
   `new Http3Transport("https://server.test/foo")`
2. Browser creates an HTTP/3 connection to `server.test` or uses existing one
3. Browser sends a CONNECT request with following headers:
   a. ":protocol" set to "webtransport"
   b. ":path" set to "/foo"
   c. ":authority" set to "server.test"
   d. "Origin" set to "https://example.com"
4. Server responds with 200 OK that has ":sessionid" header set to 1.
5. Both peers can send streams and datagrams associated with ID 1.
6. The session is closed if the associated CONNECT stream is closed.

Http3Transport open issues

- How do we fall back when HTTP/3 is not available?
- What are the differences between this and “HTTP/2 as a transport” drafts?
  - Do we want to support custom headers on the CONNECT request?
  - Do we want to support headers and/or trailers on data streams?
- Do we want to provide consistent stream ID view to client and server across HTTP proxies?
- Can we use SETTINGS instead of transport parameters?

Comparison: QuicTransport vs RTCDataChannel

	RtcDataChannel	QuicTransport
Connection model	P2P (via ICE)	Direct
Transport protocol	SCTP	QUIC
Trust model	Mutual TLS with certificate fingerprint exchanged out-of-band	Web PKI
Consent to send	Via ICE	QUIC with ALPN
Objects	Messages	Streams, datagrams
Large message support	Poor (blocks the channel without NDATA support)	Just works

Comparison: QuicTransport vs WebSocket

	WebSocket	QuicTransport
Head-of-line blocking	Always	Only inside same stream
Partial reliability	None	Datagrams, cancellable streams
Trust model	TLS, Origin header	TLS, Origin header
Preventing cross-protocol attacks	SHA-1 based handshake	ALPN
Preventing middlebox confusion	XOR-based masking scheme	n/a (always encrypted)
Authentication features	Cookies	None (up to application)

Q&A
(25 minutes)

Q&A End: 14:50

Pointer to Charter Discussion (5 minutes)

Presentation End: 14:55

Bernard Aboba

David Schinazi

Pointer to Charter Discussion

- This is a non-WG forming BOF, so we're not finalizing the charter today.
- However, there is an ongoing discussion on a potential charter on the mailing list.
- A draft charter is available on Github:
  - https://github.com/DavidSchinazi/webtrans-wg-materials/blob/master/charter.md
- If you have opinions, please post to the webtransport@ietf.org mailing list, or file Issues and PRs in the Github repo!

Charter Proposal

The WebTransport working group will define new client-server protocols or protocol extensions in order to support the development of the W3C WebTransport API https://wicg.github.io/web-transport.

These protocols will support:

- Reliable bidirectional and unidirectional communication that provides greater efficiency than Websockets (e.g. removal of head-of-line blocking).
- Unreliable datagram communication, functionality not available in Websockets.
- Origin checks to allow supporting the Web's origin-based security model.

The WebTransport working group will define three variants:

- A protocol directly running over QUIC with its own ALPN.
- A protocol that runs multiplexed with HTTP/3.
- Fallback protocols that can be used when QUIC or UDP are not available.

Charter Proposal (cont'd)

The group will pay attention to security issues arising from the above scenarios so as to ensure against creation of new modes of attack, as well as to ensure that security issues addressed in the design of Websockets remain addressed in the new work.

To assist in the coordination with W3C, the group will initially develop an overview document containing use cases and requirements in order to clarify the goals of the effort. Feedback will also be solicited at various points along the way in order to ensure the best possible match between the protocol extensions and the needs of the W3C WebTransport API. The clarity and interoperability of specifications will be confirmed via test events and hackathons.

The group will also coordinate with other working groups within the IETF (e.g. QUIC, HTTPBIS) as appropriate.

Charter Proposal (cont'd)

Goals and Milestones

- March 2020 - Adopt a WebTransport Overview draft as a WG work item
- March 2020 - Adopt a draft on WebTransport over QUIC as a WG work item
- March 2020 - Adopt a draft on WebTransport over HTTP/3 as a WG work item
- March 2020 - Adopt a draft on HTTP/2 fallback mechanism as a WG work item
- March 2020 - Adopt a draft on a QUIC fallback mechanism as a WG work item
- August 2020 - Issue WG last call of the WebTransport Overview document.
- November 2020 - Issue WG last call on WebTransport over QUIC
- November 2020 - Issue WG last call on QUIC fallback mechanism
- February 2021 - Issue WG last call on WebTransport over HTTP/3
- February 2021 - Issue WG last call on HTTP/2 fallback mechanism

Wrapup and Summary (5 minutes)

Session End: 15:00

Bernard Aboba

David Schinazi

Questions

1) Is the WebTransport problem statement clear, well-scoped, solvable, and useful to solve?

2) Are the WebTransport deliverables (WebTransport overview, QuicTransport, Http3Transport, FallbackTransport) well-defined and well-understood?

3) Are you willing to review documents (and/or comment on the mailing list)?

WEBTRANS BOF
IETF 106

Singapore

Wednesday, November 20, 2019

13:30 – 15:00

Room: Collyer

Meetecho: https://www.meetecho.com/ietf106/webtrans/
Mailing list: webtransport@ietf.org
Jabber Room: webtrans@jabber.ietf.org