Using HTTP/2 as a Transport for Arbitrary Bytestreams

Eric Kinnear (ekinnear@apple.com)
Tommy Pauly (tpauly@apple.com)

TSVWG
IETF 104, March 2019, Prague
Motivation

Generic transport for secure, arbitrary bytestreams

Multiplexed streams

  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 from TLS/TCP
Potential Solution

CONNECT allows tunneling to another endpoint
Extended CONNECT allows connecting to server itself
Can also enable proxying of UDP, with additional framing
HTTP headers enable additional negotiation
Coexists with standard HTTP request/response streams
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 values

“bytestream”

Direct stream mapping for arbitrary bytestreams to remote server

“datagram”

Framing for UDP transport, to server and possibly with traditional CONNECT to another endpoint
Motivation

Generic transport for secure, arbitrary bytestreams

Multiplexed streams

- 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, arbitrary bytestreams

Multiplexed streams
  - 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 Transport?

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

What transport abstraction does QUIC Transport 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

Define additional :protocol values

“bytestream”

Direct stream mapping for arbitrary bytestreams to remote server

“datagram”

Framing for UDP transport, to server and possibly with traditional CONNECT to another endpoint

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

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 Transport over HTTP/2 framing
Transport Properties

Sharing underlying transport brings benefits, but also has caveats

- Head-of-Line blocking
- Connection limits on flow control
- Additional items?

Other work on multiplexed transports (SCTP)

- How much to reference or include?
Questions?