Transport Services API for QUIC

draft-pauly-quic-interface-00

Tommy Pauly, Eric Kinnear
TAPS
IETF 103, November 2018, Bangkok
Goals for a QUIC API

Make it easy to transition clients to QUIC

Racing and fallback provided by implementation

Common API surface between QUIC and TCP-based solutions

Expose novel transport features (0-RTT, multistreaming)
Transitioning Clients to QUIC

HTTP Application

HTTP API

TLS

TCP

IP

QUIC

UDP

IP
Transitioning Clients to QUIC

Foo Application

TLS/DTLS
TCP/UDP
IP

QUIC
UDP
IP
Transitioning Clients to QUIC

Foo Application

TAPS API

TLS
TCP
IP

QUIC
UDP
IP
Transitioning Clients to QUIC

HTTP Application

HTTP API

TAPS API

TLS

TCP

IP

QUIC

UDP

IP
Models for using QUIC
As a TCP-style stream

Use only one stream in a connection

Benefit from:

• Faster secure handshake
• 0-RTT support
• Path migration
• Extensible transport parameters
• More complete privacy and authentication
Models for using QUIC
As a message channel

Consume a stream for each application message

Benefit from:

• All connection-level benefits
• Eliminating head-of-line blocking between messages
• Explicit uni-directional vs bi-directional streams provide “reply” semantics
Models for using QUIC
As a tunnel for several streams

Use many long-lived streams

Benefit from:

- Shared security and authentication context between many streams
- Unreliable extensions can allow parallel unreliable and reliable data within a connection
“Stream” Mode
Transport connection as QUIC stream

- Initiate()
- Send(partial)
- Send(complete)
- Receive()
- Close()

QUIC Handshake
New Stream
Send STREAM
Receive STREAM
STREAM+FIN
RST_STREAM
CONNECTION_CLOSE
“Connection” Mode
Transport connection as QUIC connection

Initiate() → QUIC Handshake
Send(partial) → New Stream
Send(complete) → Send STREAM, Receive STREAM
Receive() → STREAM+FIN
Close() → RST_STREAM, CONNECTION_CLOSE
Implementation Considerations

Even if both models may be presented to an API client, implementation should be the same.

“Stream” mode has more complete expressivity.

One caveat is implicit stream opening.

Generally, build “Connection” on top of “Stream”.

Same two models can be used for other multi-streaming transports, like HTTP/2.