QUIC Unidirectional and Bidirectional Streams

IETF 99

By Ian Swett

Background

QUIC needs to support HTTP

HTTP/2 supports both request/response and server push

Therefore the transport and application doc COMBINED must provide BOTH bidirectional streams and unidirectional streams

Background

QUIC needs to support HTTP

HTTP/2 supports both request/response and server push

Therefore the transport and application doc COMBINED must provide BOTH bidirectional streams and unidirectional streams

Use only one direction of a bidirectional stream

OR

Build bidirectional streams on two unidirectional streams

Bidirectional: Current draft

Pairs of incoming and outgoing streams: Think TCP

Pros:

- Ideal for HTTP requests and responses
- Large amounts of deployment experience

Cons:

- Requires sending an unnecessary STREAM frame with a FIN for server push
- Bugs:

Bidirectional: With Application manipulation (gQUIC)

Bidirectional, but the application closes half the stream without sending a FIN

Pros:

- Ideal for HTTP requests
- Large amounts of deployment experience
- Doesn’t require sending an unnecessary FIN

Cons:

- Transport state requires knowledge of the application state
- Feels a bit ugly to allow manipulating internal transport state

Bidirectional: Open streams half-closed

Adds a bit in the STREAM frame to indicate it should be opened half closed.

Pros:

- Saves sending an unnecessary response FIN for Server Push
- Small change to the existing draft

Cons:

- Uses a bit in the type byte
- Adds some transport complexity, particularly around implicitly opened streams
- No deployment experience

PR #656

Unidirectional Only in Transport

All streams are unidirectional. Pair two unidirectional streams to create bidirectional streams. Stream IDs are available on both sides.

> *Example: Can create a bidirectional stream from 5 on the client side and 5 on the server side or 5 on the client side and 18 on the server side.*

PR #643

Unidirectional Only in Transport (continued)

Pros:

- Simplifies the transport document
- Simplifies server push
- More generic - applications can choose their own model

Cons:

- Makes the HTTP mapping doc more complex
- Each application that needs request/response mapping will need to create it.
- MAX_STREAM limits can allow sending a request, but not a response.
- No deployment experience

PR #643

Unidirectional plus a binding layer

Unidirectional only, plus a means of associating two streams in the transport

Pros:

- More capable than fully unidirectional
- Allows many-to-one

Cons:

- Less general than fully unidirectional
- More complex than unidirectional or bidirectional
- Allows many-to-one
- No deployment experience

PR #672

Points to consider

- Server push is rare: <1% of responses
- Sticking with the current draft is very much an option
- Only plain bidirectional streams have implementation and deployment
- Quartc uses streams in a unidirectional way heavily, so we'll have useful implementation experience 'soon'

Any change now may be premature optimization