Connection Pooling
for request/response style applications

Philipp S. Tiesel
TAPS
IETF 104, March 2019, Prague
Connection Pooling

- Alternate API interaction scheme for request/response style applications.
- Combine several underlying transport connections into one pooled connection.
- Automated initiation and teardown of additional underlying connections.
- Match request and responses through (local) message references.
API for QUIC

form draft-pauly-quic-interface-00

https://datatracker.ietf.org/meeting/103/materials/slides-103-taps-3a-taps-api-mappings-for-quic-00
“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

https://datatracker.ietf.org/meeting/103/materials/slides-103-taps-3a-taps-api-mappings-for-quic-00
API for QUIC

form draft-pauly-quic-interface-00

https://datatracker.ietf.org/meeting/103/materials/slides-103-taps-3a-taps-api-mappings-for-quic-00

“Stream” Mode
Transport connection as QUIC stream

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

“Connection” Mode
Transport connection as QUIC connection

- Initiate()
- Send(partial)
- Send(complete)
- Receive()
- Close()
API for QUIC

form draft-pauly-quic-interface-00
https://datatracker.ietf.org/meeting/103/materials/slides-103-taps-3a-taps-api-mappings-for-quic-00

“Stream” Mode
Transport connection as QUIC stream

Initiate() -> QUIC Handshake
Send(partial) -> New Stream
Send(complete) -> Send STREAM, Receive STREAM
Receive() -> STREAM+FIN
Close() -> 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

Regular Connection
API for QUIC

form draft-pauly-quic-interface-00
https://datatracker.ietf.org/meeting/103/materials/slides-103-taps-3a-taps-api-mappings-for-quic-00

"Stream" Mode
Transport connection as QUIC stream
- Initiate()
- Send(partial)
- Send(complete)
- Receive()
- Close()

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

"Connection" Mode
Transport connection as QUIC connection
- Initiate()
- Send(partial)
- Send(complete)
- Receive()
- Close()

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

Regular Connection

Pooled Connection
Connection Pooling

Functionality

- Same API for HTTP/1.1, HTTP/2, and HTTP/3, allows mixing them transparently.
- Automatic management of QUIC Streams for HTTP/3.
- Replacement for unbound UDP sockets.
- Replacement for the “one-to-many” interface of SCTP.
- Enable transparent connection migration.
- Enable per-message path selection.
Connection Pooling Variants

- Add Connection Pool Object – PR #295
  - Separate Connection Pool object.
  - A TAPS Connections always represents one underlaying transport connection.

- Add Pooled Connections – PR #298
  - Selection Property enables Pooled Connections.
  - A TAPS connection can represent multiple underlaying transport connections.
Connection Pool Example

```plaintext
RemoteSpecifier := NewRemoteEndpoint()
RemoteSpecifier.WithHostname("example.com")
RemoteSpecifier.WithService("https")
TransportProperties := NewTransportProperties()
TransportProperties.Require(preserve-msg-boundaries)
TransportProperties.Ignore(preserve-order)
// Security Parameters left out for brevity

Preconnection := NewPreconnection(None, RemoteSpecifier, TransportPreperties, SecurityParameters)
RequestorPool := Preconnection.RequestorPool()

// no ready event
reqRef := RequestorPool.Send(messageData: Request, reqRef: None)

RequestorPool.Receive()
RequestorPool -> Received(messageDataResponse, messageContext, requestRef)

RequestorPool.Stop()
```
Pooled Connection Example

RemoteSpecifier := NewRemoteEndpoint()
RemoteSpecifier.WithHostname("example.com")
RemoteSpecifier.WithService("https")
TransportProperties := NewTransportProperties()
TransportProperties.Require(preserve-msg-boundaries)
TransportProperties.Ignore(preserve-order)
TransportProperties.Prefer(pool-connections)
// Security Parameters left out for brevity

Preconnection := NewPreconnection(None, RemoteSpecifier, TransportProps, SecurityParameters)
Connection := Preconnection.Initiate()

Connection -> Ready<>  
reqRef := Connection.Send(messageData: Request, reqRef: None)

Connection.Receive()
Connection -> Received(messageDataResponse, messageContext, reqRef)

Connection.Close()