UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

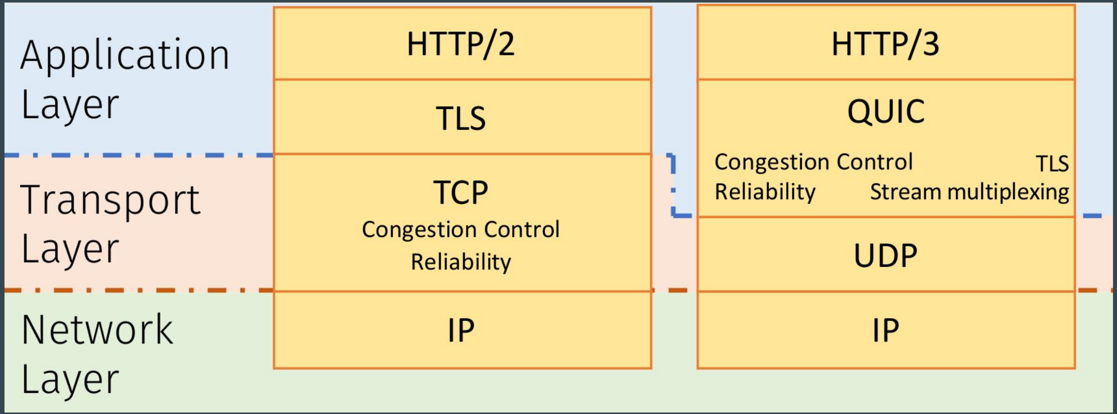
A QUIC implementation for ns-3

Alvise De Blasio, **Federico Chiariotti**, Michele Polese, Andrea Zanella, Michele Zorzi

https://github.com/signetlabdei/quic-ns-3
https://apps.nsnam.org/app/quic/

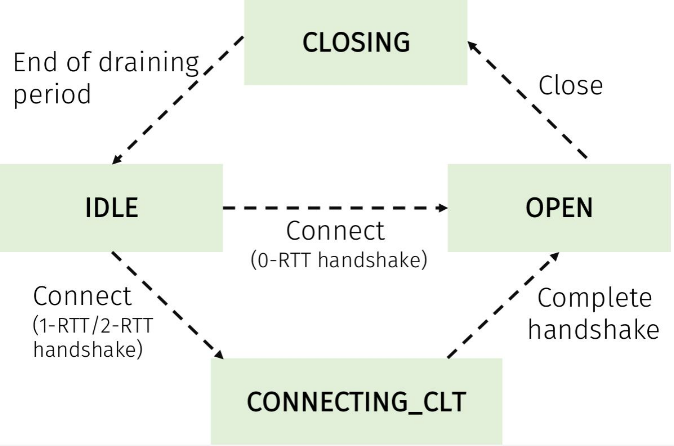
The QUIC protocol

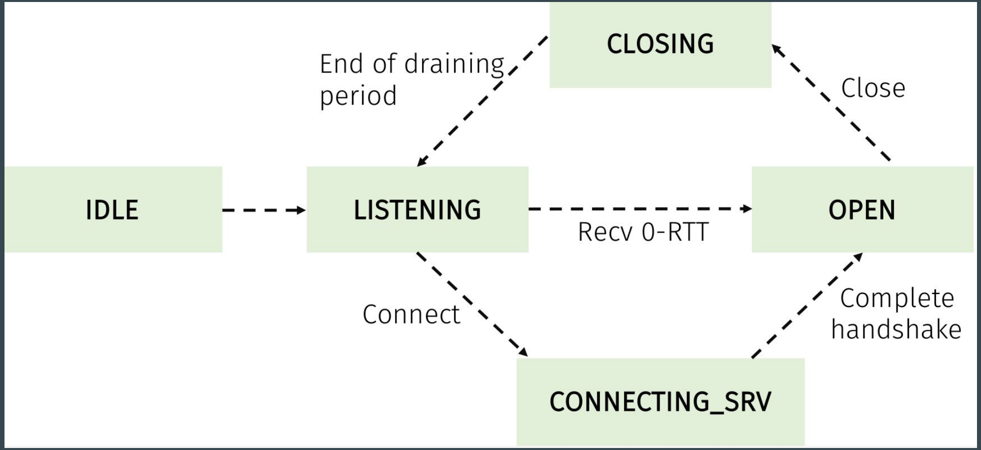
- Developed by Google in 2013, currently used for 30% of Google traffic
- IETF Internet Draft (ongoing standardization process)
- Runs over UDP and integrates TLS 1.3 support
- Native SACK support, better RTT estimation



Connection setup

- 0-RTT: one-way packet from the client (for previously established pairs)
- 1-RTT: TCP-like handshake with TLS parameter negotiation
- 2-RTT: Version negotiation, then 1-RTT handshake



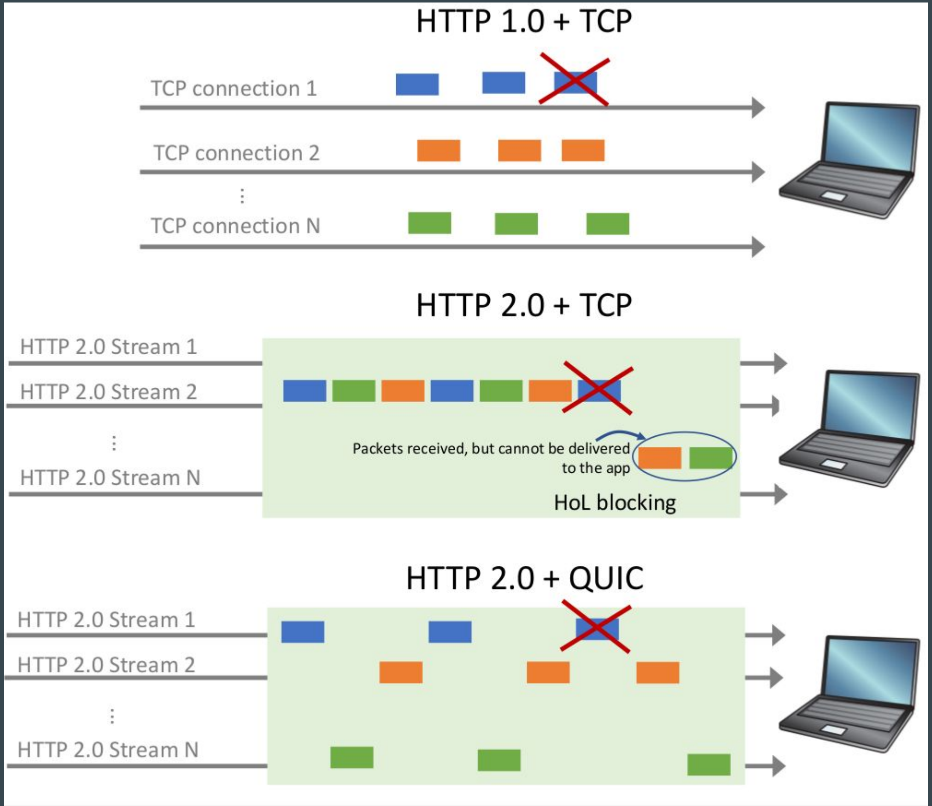


QUIC streams and HTTP

1. HTTP/1 opens a different TCP connection for every object, with a separate congestion control

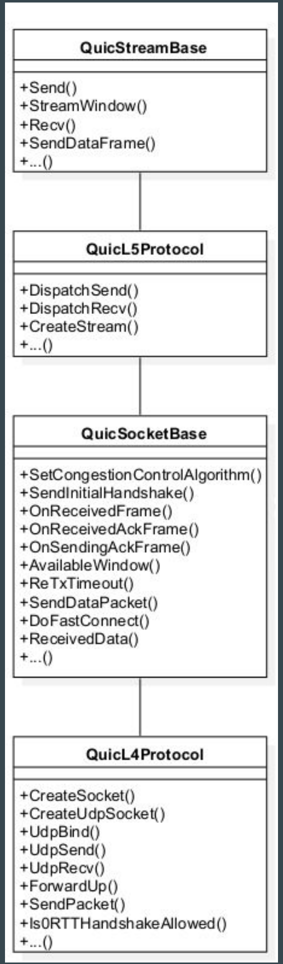
2. HTTP/2 uses the same TCP connection, but packet loss for the first object can block subsequent ones

3. QUIC requires in-order delivery on a stream level, so HoL is prevented (HTTP/3)



The QUIC ns-3 module

1. Inherits the logic of the TCP implementation
2. The `QuicSocketBase` class performs the basic socket functions
3. The `QuicL4Protocol` class handles interactions with the underlying UDP socket
4. The `QuicL5Protocol` class manages streams
5. Basic stream functions are performed by the `QuicStreamBase` class



QuicStreamBase

+Send()
+StreamWindow()
+Recv()
+SendDataFrame()
+...()

QuicL5Protocol

+DispatchSend()
+DispatchRecv()
+CreateStream()
+...()

QuicSocketBase

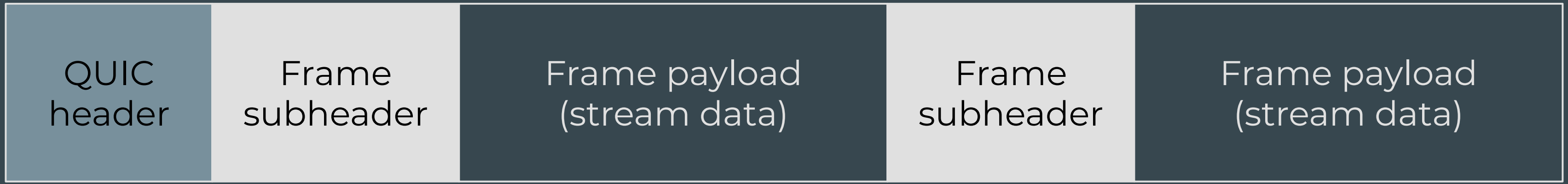
+SetCongestionControlAlgorithm()
+SendInitialHandshake()
+OnReceivedFrame()
+OnReceivedAckFrame()
+OnSendingAckFrame()
+AvailableWindow()
+ReTxTimeout()
+SendDataPacket()
+DoFastConnect()
+ReceivedData()
+...()

QuicL4Protocol

+CreateSocket()
+CreateUdpSocket()
+UdpBind()
+UdpSend()
+UdpRecv()
+ForwardUp()
+SendPacket()
+Is0RTTHandshakeAllowed()
+...()

QUIC packet structure

- Encapsulated into a UDP datagram
- The `QuicHeader` class implements the header
- Headers can be long (17 B, used in the connection setup) or short (2-13 B)
- The `QuicSubheader` class implements the frame subheader
- Data frames are associated to streams, control frames have a custom format



QUIC header

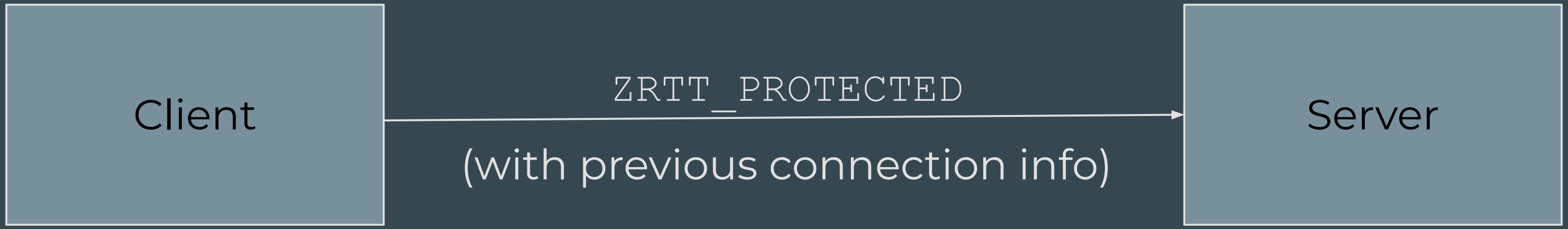
Frame subheader

Frame payload (stream data)

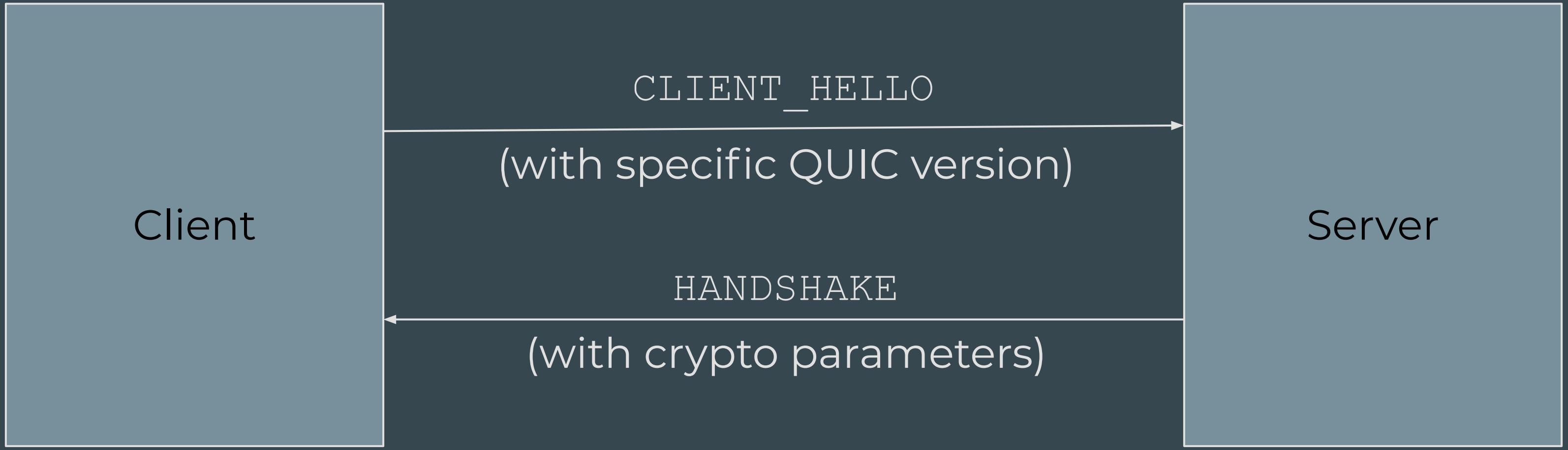
Frame subheader

Frame payload (stream data)

Connection setup: 0-RTT



Connection setup: 1-RTT



Client

CLIENT_HELLO

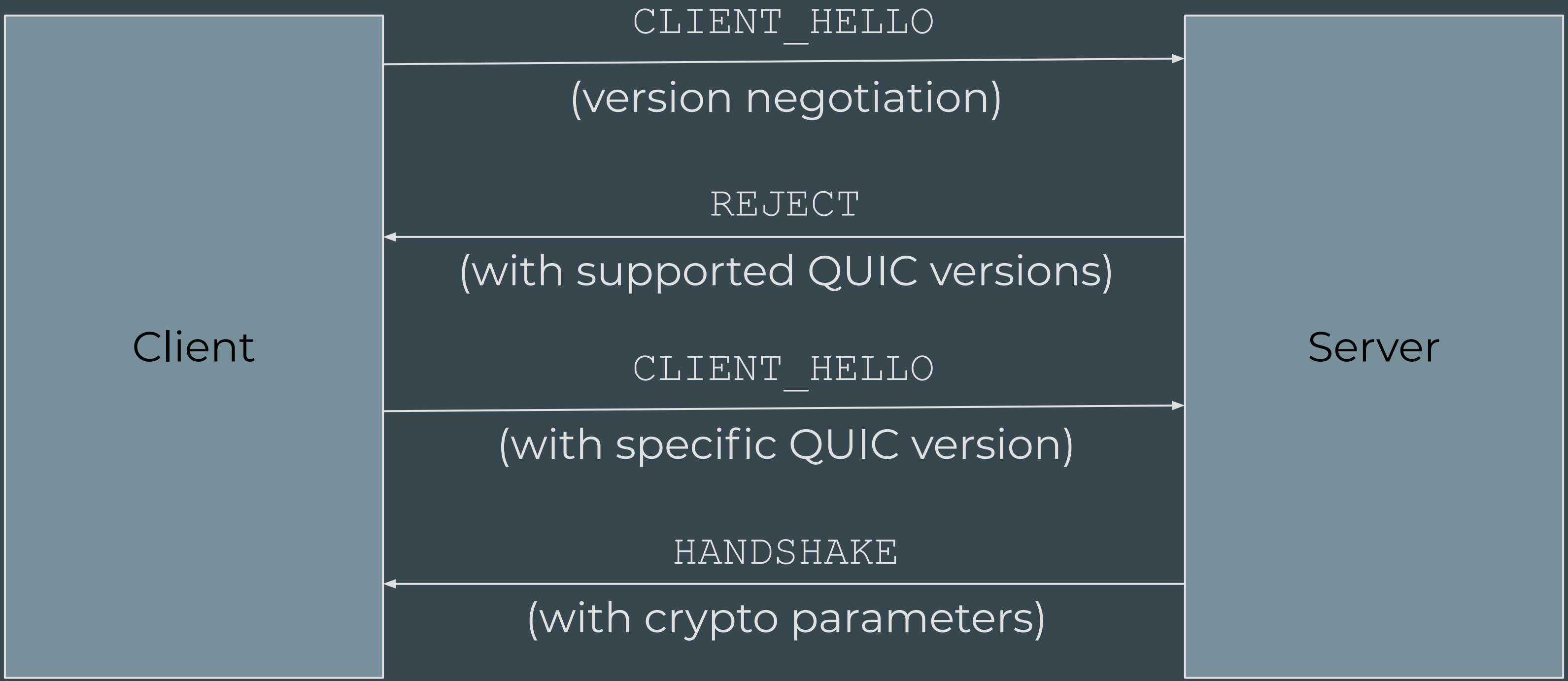
(with specific QUIC version)

Server

HANDSHAKE

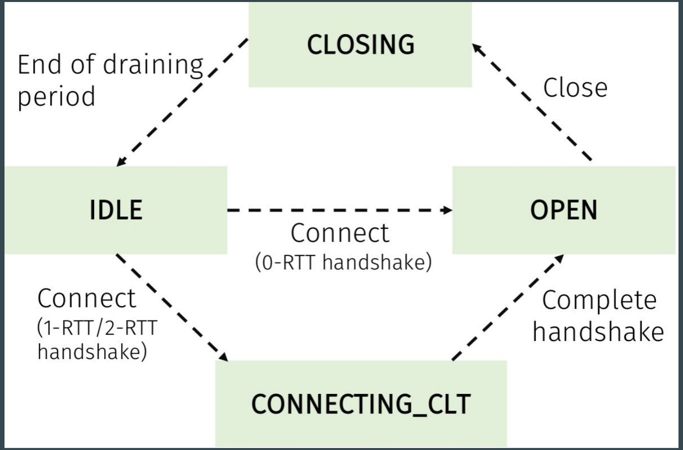
(with crypto parameters)

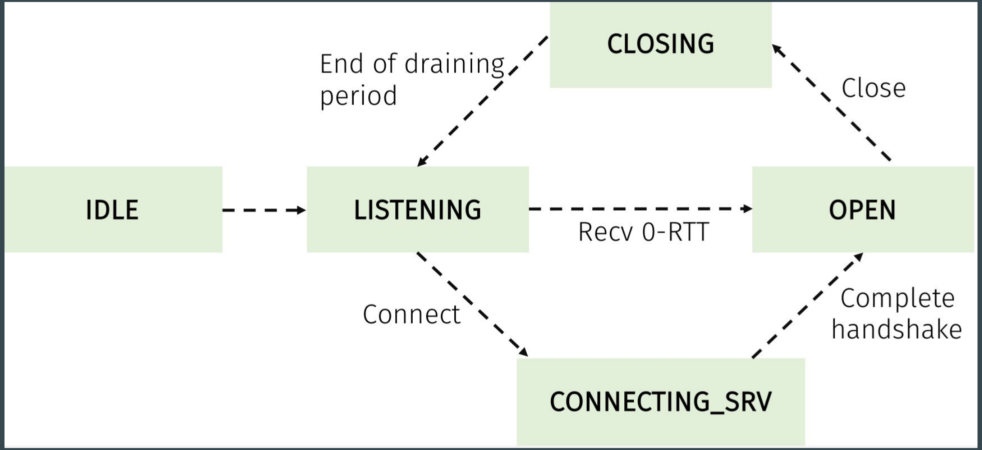
Connection setup: 2-RTT



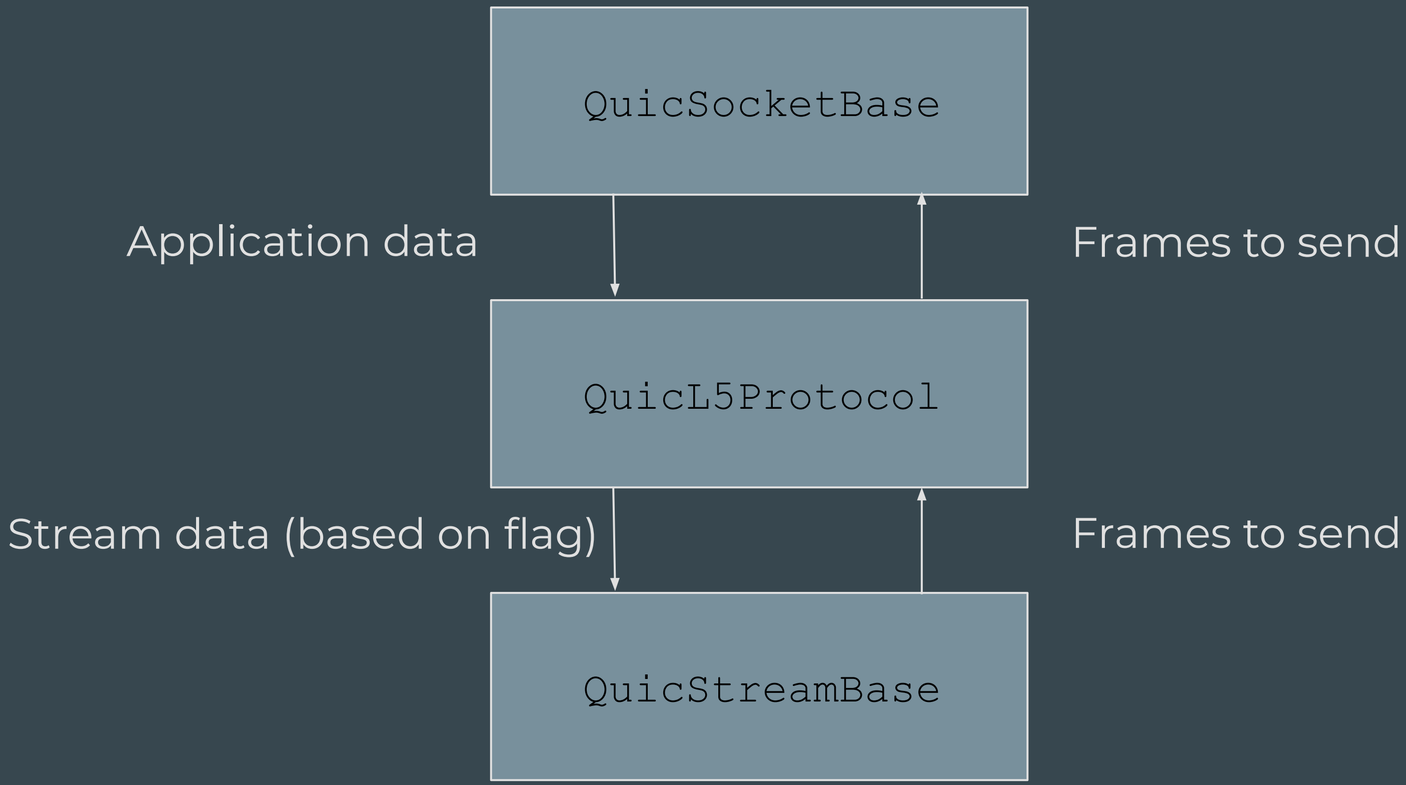
Connection setup in ns-3

- Simulated TLS handshake
- No need for external crypto libraries





Data flow in the QUIC module (sender)



QuicSocketBase

Application data

Frames to send

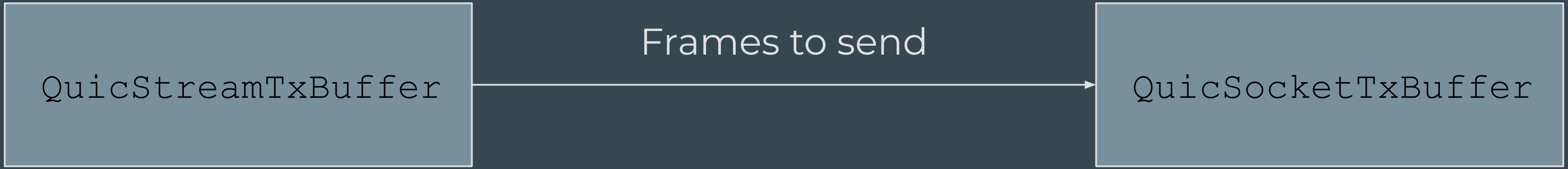
QuicL5Protocol

Stream data (based on flag)

Frames to send

QuicStreamBase

QUIC: send buffers



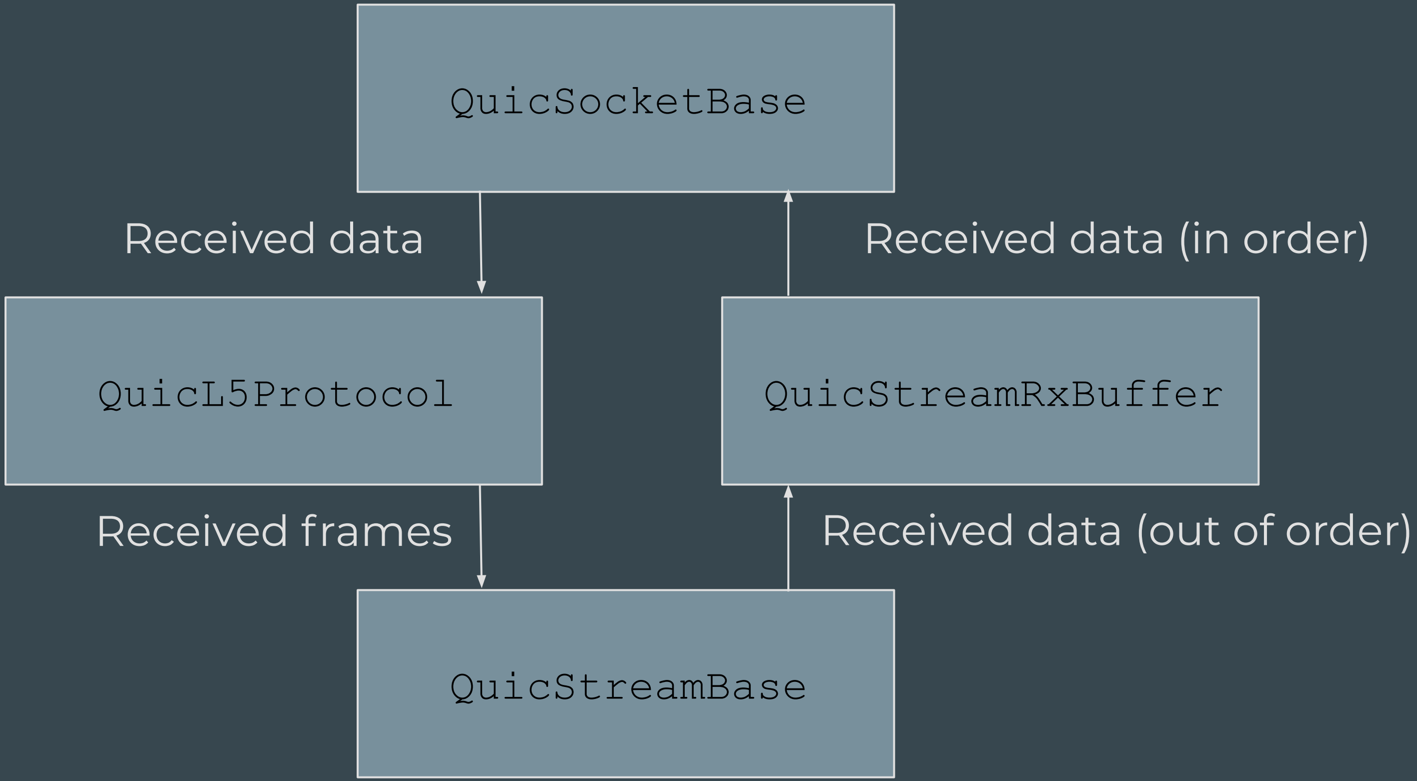
`QuicStreamTxBuffer`

Frames to send

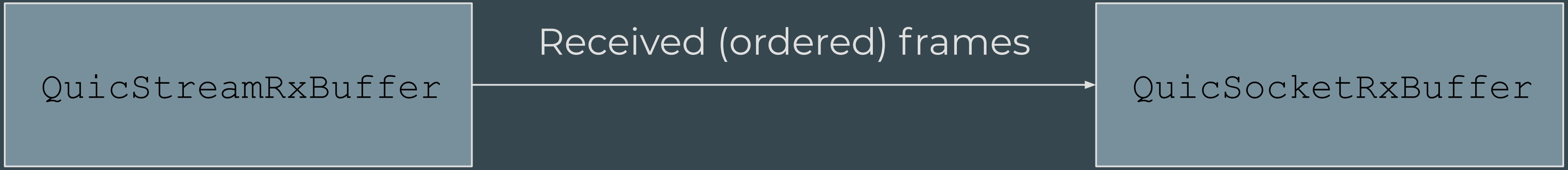
`QuicSocketTxBuffer`

- The socket buffer has a list of sent and waiting packets
- Stream 0 (control frames) frames are sent with high priority
- Retransmissions and ACKs are handled by the socket buffer
- The stream buffer stores packets and avoids socket buffer overflows

Data flow in the QUIC module (receiver)



QUIC: receive buffers



- The socket disgregates received packets and passes frames to the stream
- The stream buffer handles reordering (for each stream)
- In-order bytes are written to the socket buffer
- The application reads a bytestream from the socket buffer

Congestion control

- Legacy mode: use TCP congestion control
- `SetCongestionControlAlgorithm` accepts any class that extends `TcpCongestionOps`

- QUIC draft mode: use QUIC-specific congestion control
- The `QuicCongestionOps` class extends `TcpNewReno`
- Full support for the QUIC Internet Draft specification

QUIC congestion control

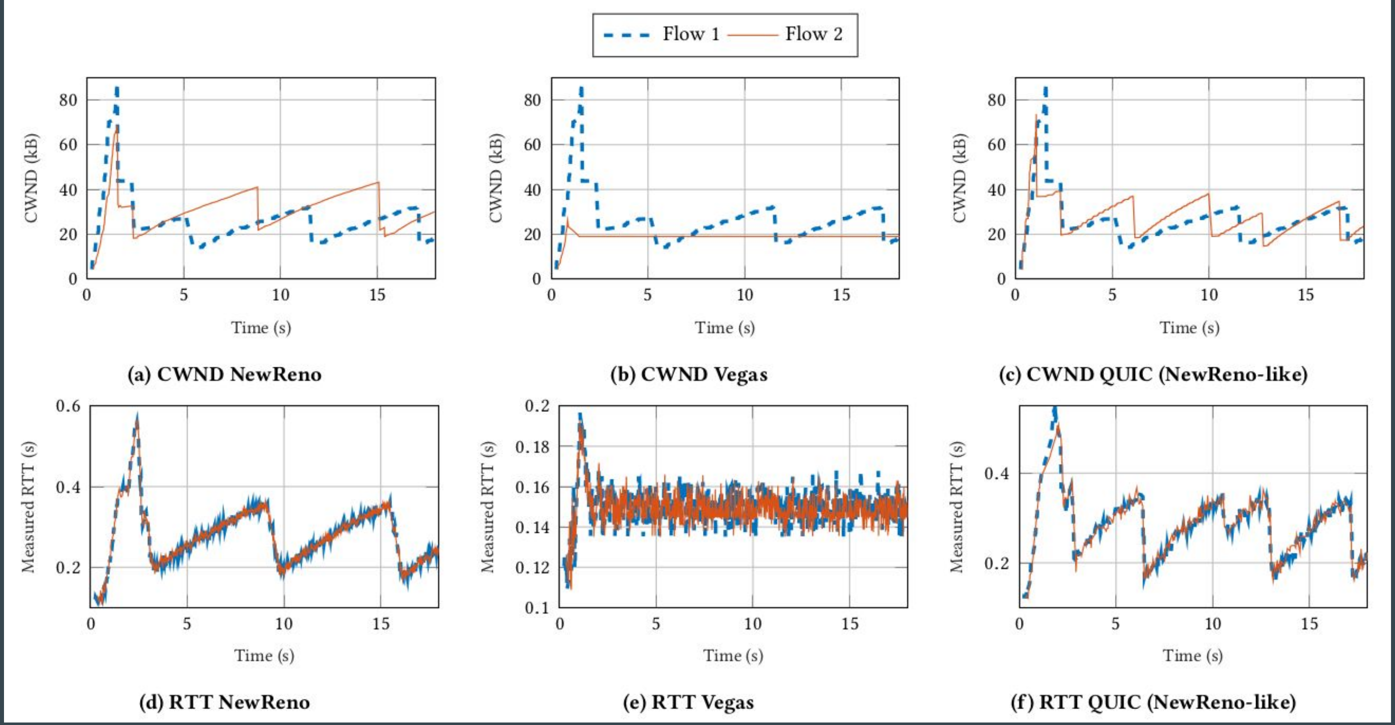
- ➔ Better RTT estimation (explicit receiver-side delay signaling)
- ➔ Retransmitted packets have a different sequence number
- ➔ Optional short loss timer (counted as DUPACK)

Congestion control - example

C = 2 Mb/s

RTT = 100 ms

BDP ~ 50 kB



(a) CWND NewReno

(b) CWND Vegas

(c) CWND QUIC (NewReno-like)

(d) RTT NewReno

(e) RTT Vegas

(f) RTT QUIC (NewReno-like)

Future work

- Alignment with Release 18 of the QUIC IETF Draft
- Integration with BBR congestion control
- Extended unit tests and full special frame support
- Development of HTTP/3 traffic models

Thanks for your attention!

GitHub repository: https://github.com/signetlabdei/quic-ns-3

ns-3 app store: https://apps.nsnam.org/app/quic/

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE