QUIC Loss Detection

draft-ietf-quic-recovery-19

TCPM @ IETF 104, March 2019

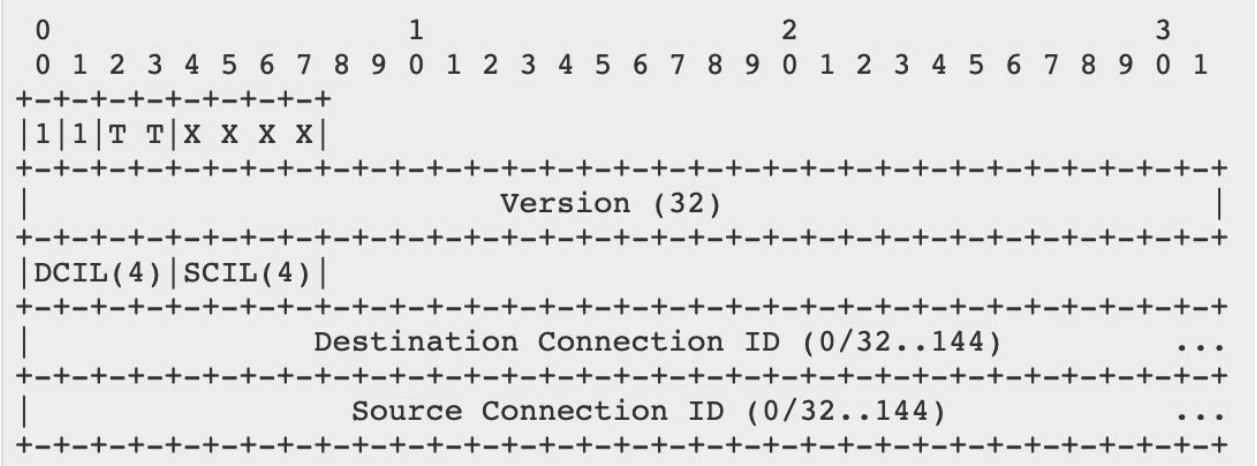
QUIC Packet Format

Long header

Short header

QUIC Packet Format

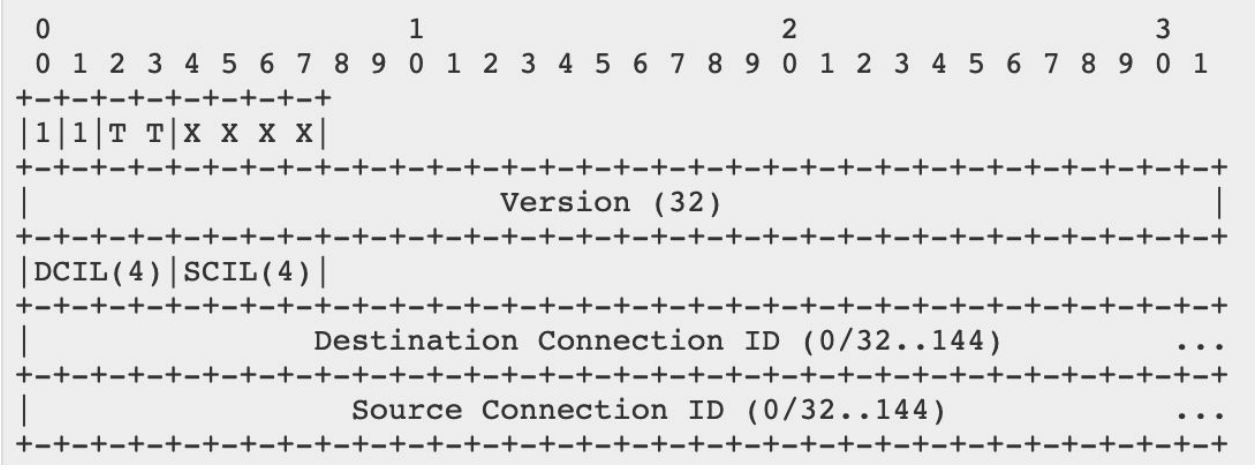
Long header



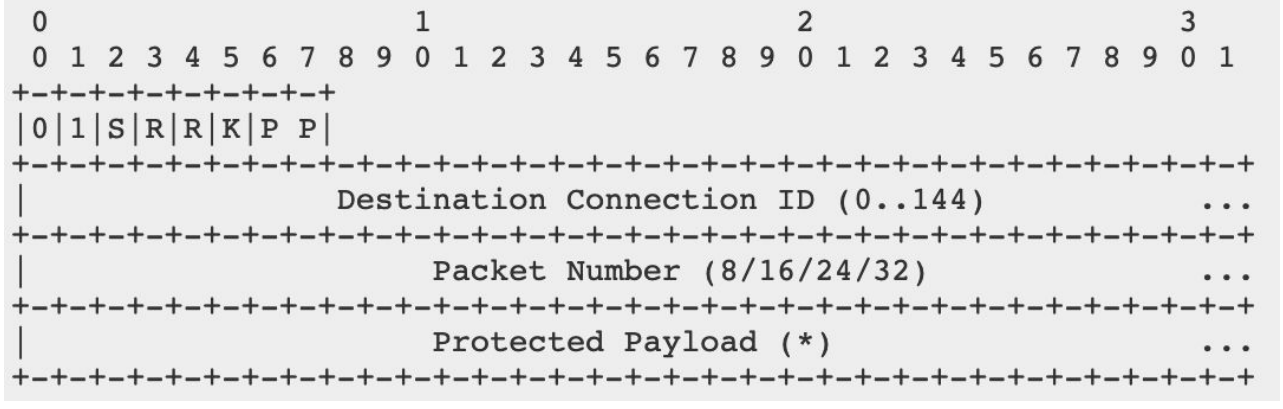
Short header

QUIC Packet Format

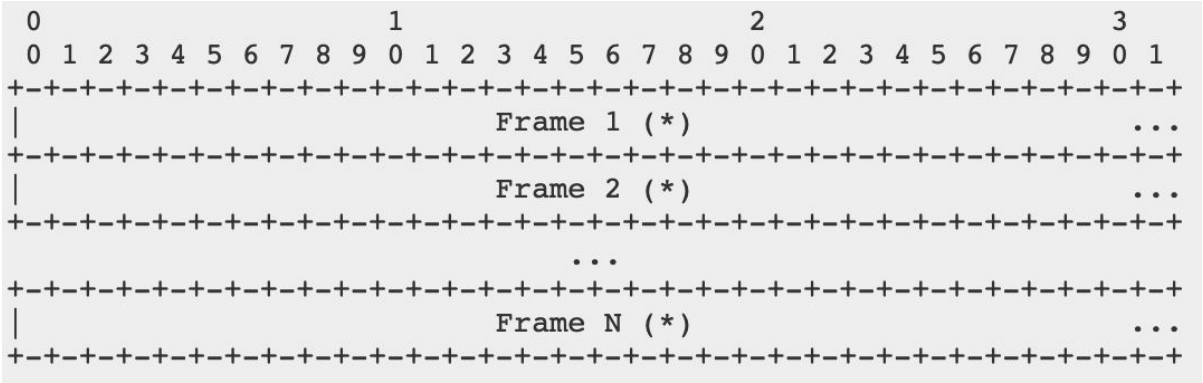
Long header



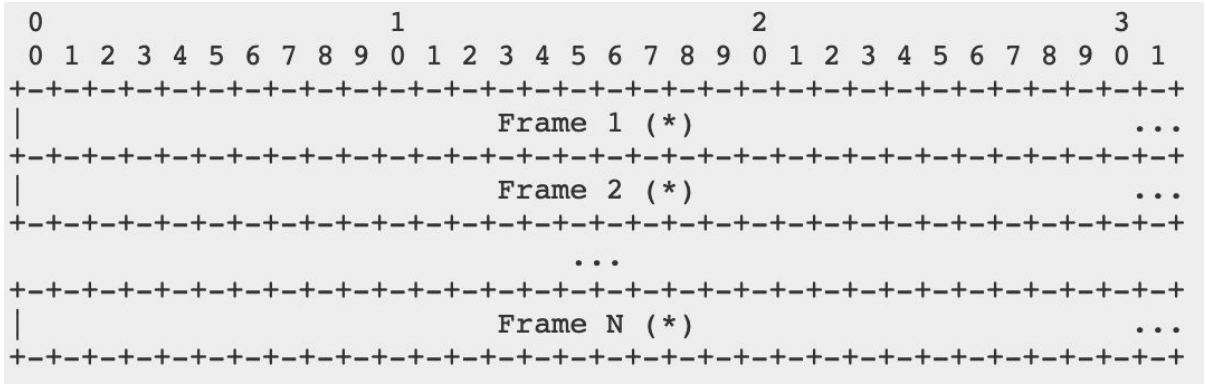
Short header

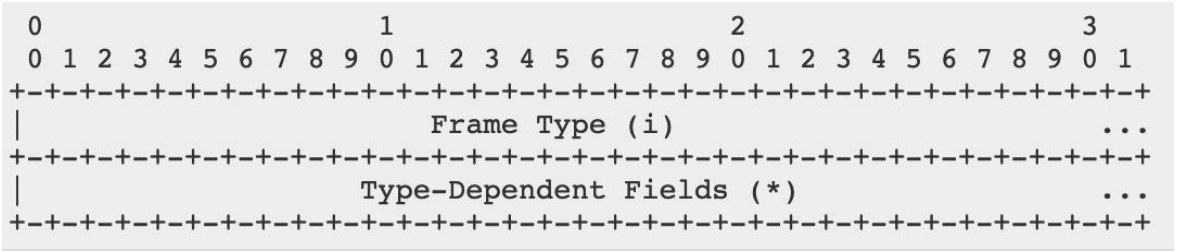


Frames

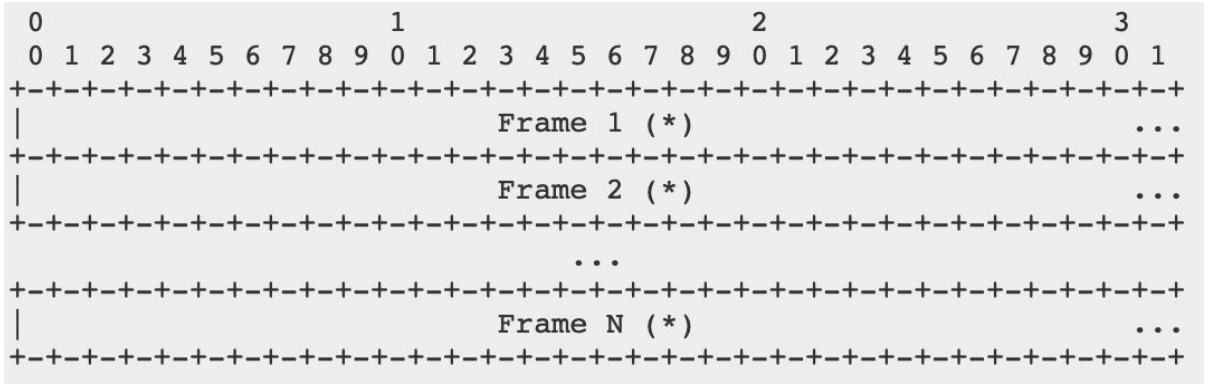

0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Frame 1 (*)
...
Frame 2 (*)
...
...
Frame N (*)
...

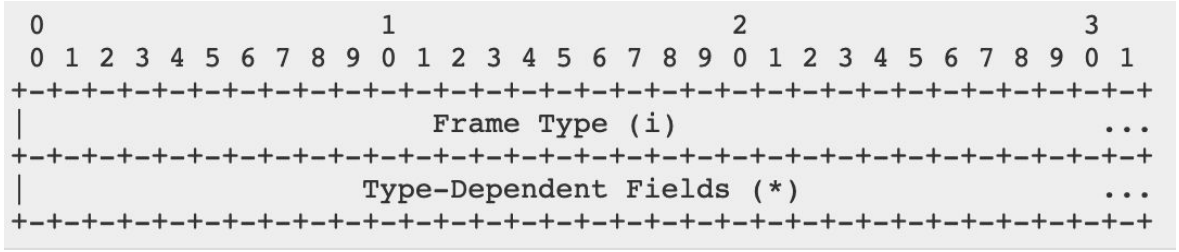
Frames


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Frame 1 (*)
Frame 2 (*)
...
Frame N (*)


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Frame Type (i)
Type-Dependent Fields (*)

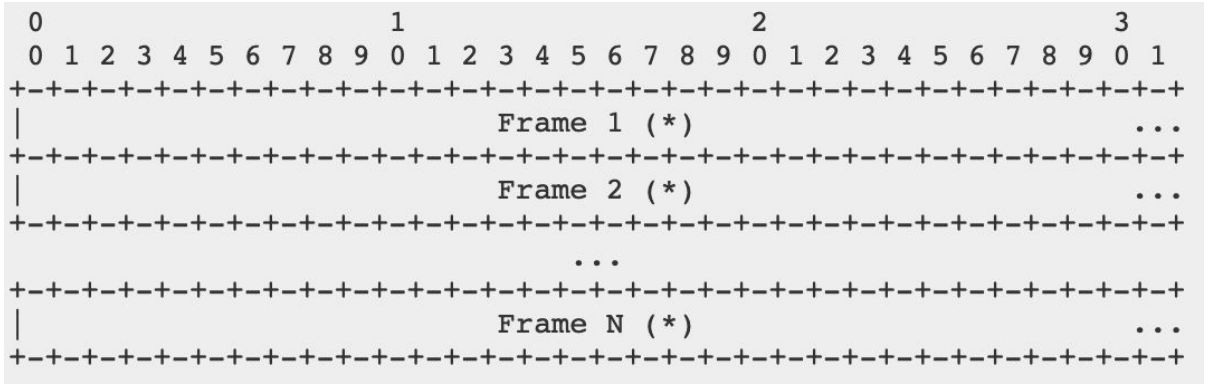
Frames

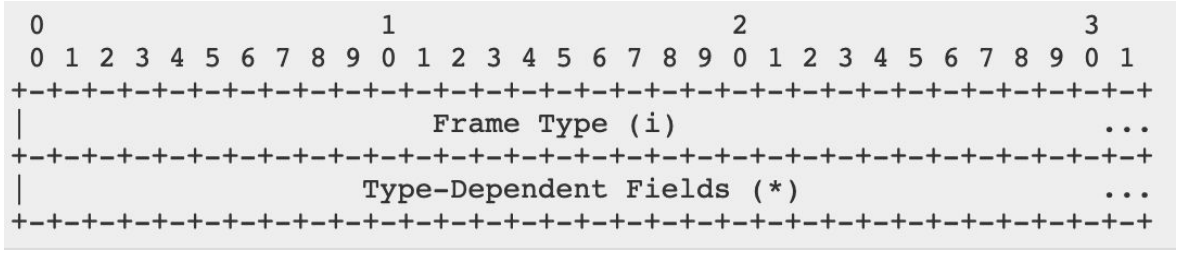




Type Value	Frame Type Name
0x00	PADDING
0x01	PING
0x02 - 0x03	ACK
0x04	RESET_STREAM
0x05	STOP_SENDING
0x06	CRYPTO
0x07	NEW_TOKEN
0x08 - 0x0f	STREAM
0x10	MAX_DATA
0x11	MAX_STREAM_DATA
0x12 - 0x13	MAX_STREAMS
0x14	DATA_BLOCKED
0x15	STREAM_DATA_BLOCKED
0x16 - 0x17	STREAMS_BLOCKED
0x18	NEW_CONNECTION_ID
0x19	RETIRE_CONNECTION_ID
0x1a	PATH_CHALLENGE
0x1b	PATH_RESPONSE
0x1c - 0x1d	CONNECTION_CLOSE

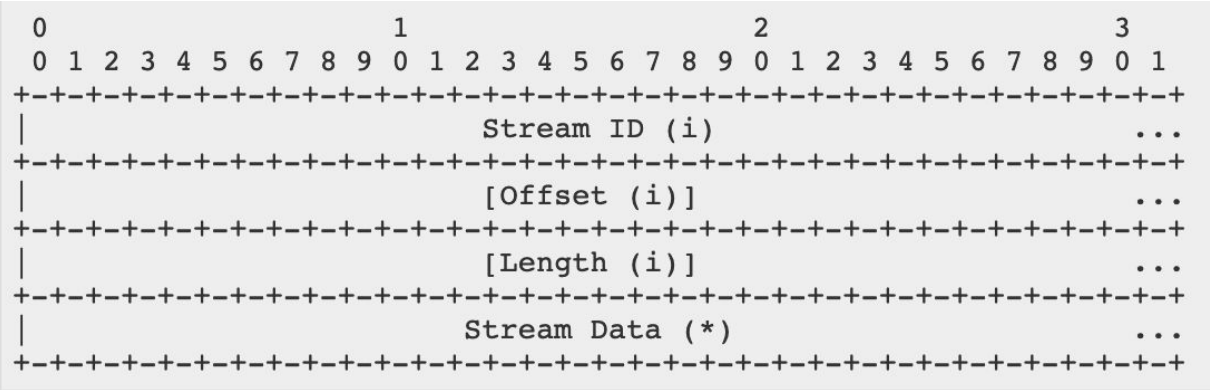
Frames



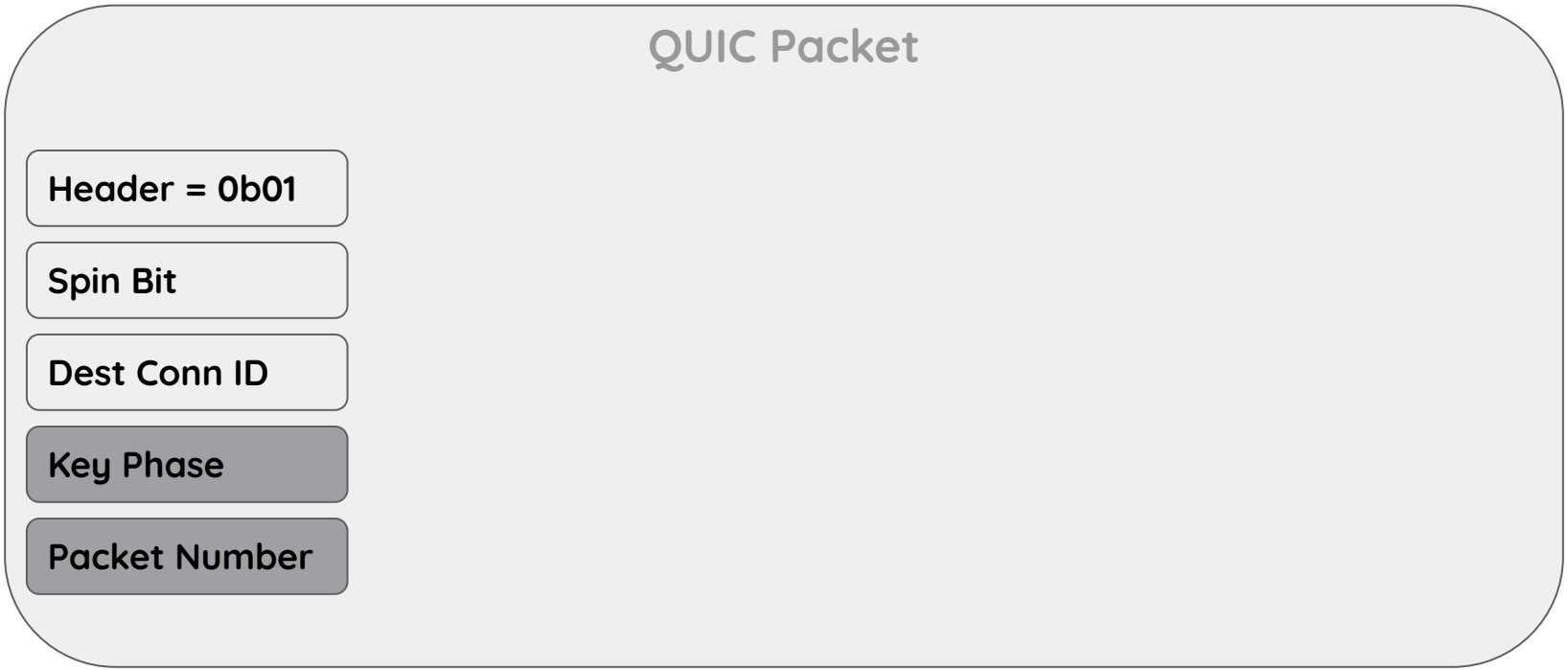


Type Value	Frame Type Name
0x00	PADDING
0x01	PING
0x02 - 0x03	ACK
0x04	RESET_STREAM
0x05	STOP_SENDING
0x06	CRYPTO
0x07	NEW_TOKEN
0x08 - 0x0f	STREAM
0x10	MAX_DATA
0x11	MAX_STREAM_DATA
0x12 - 0x13	MAX_STREAMS
0x14	DATA_BLOCKED
0x15	STREAM_DATA_BLOCKED
0x16 - 0x17	STREAMS_BLOCKED
0x18	NEW_CONNECTION_ID
0x19	RETIRE_CONNECTION_ID
0x1a	PATH_CHALLENGE
0x1b	PATH_RESPONSE
0x1c - 0x1d	CONNECTION_CLOSE

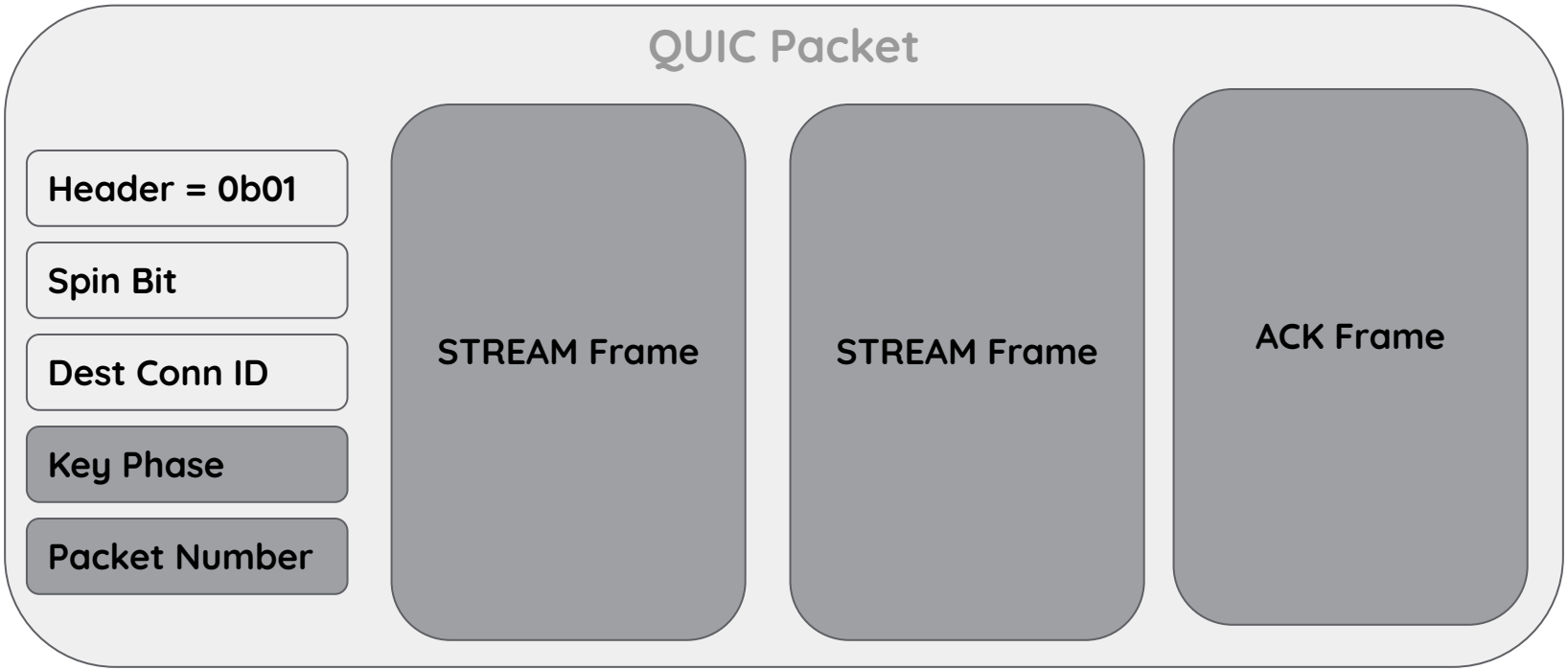
STREAM Frame



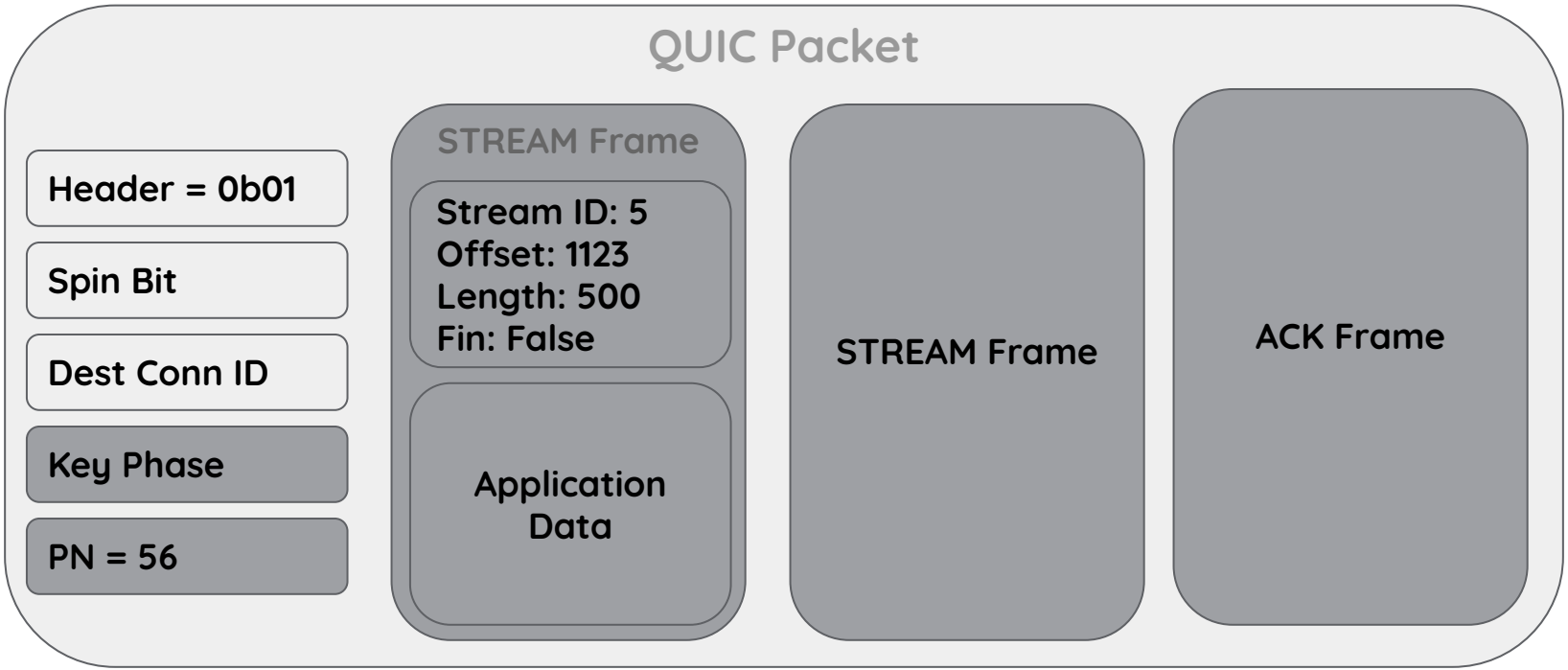
QUIC Packetization: Example



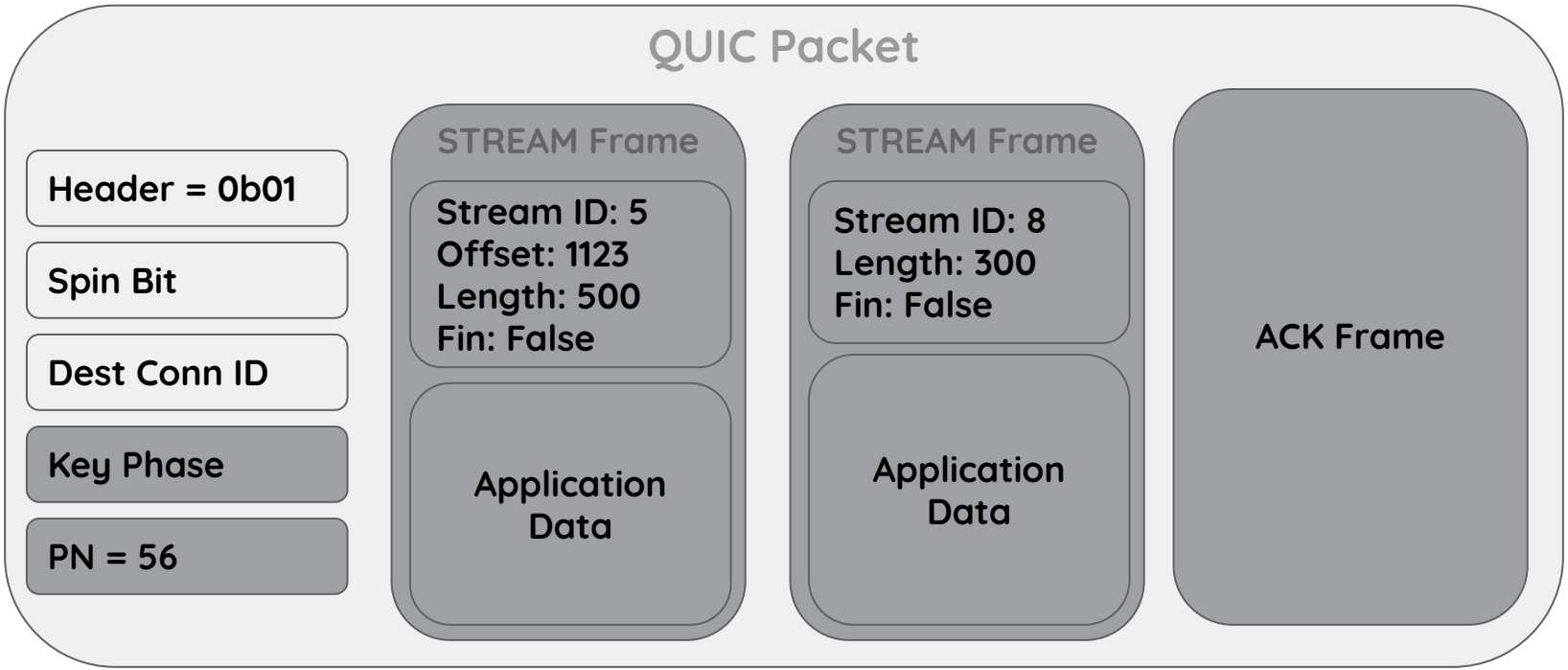
QUIC Packetization: Example



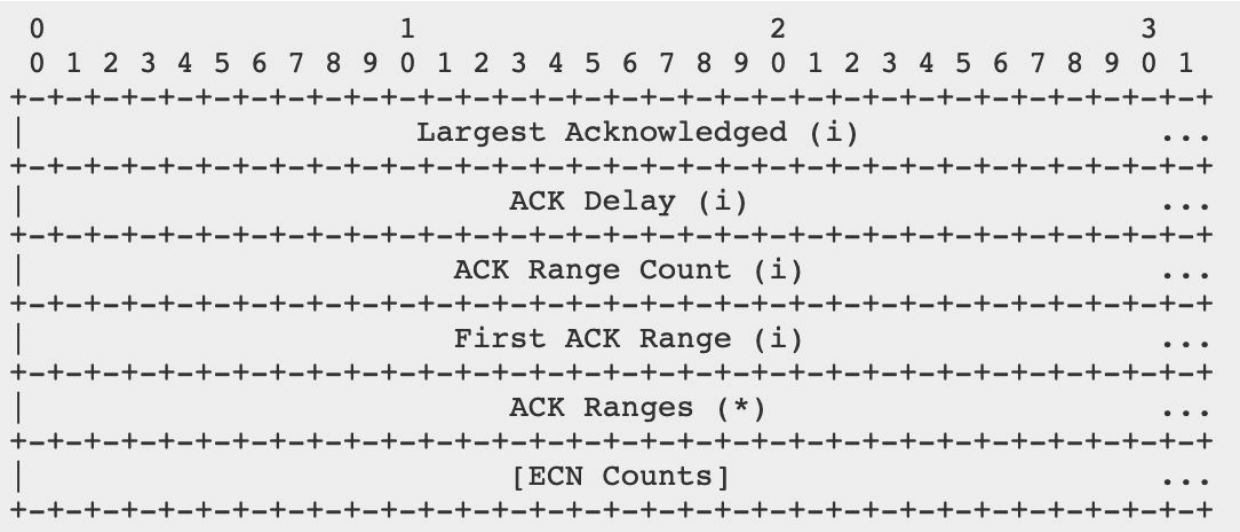
QUIC Packetization: Example



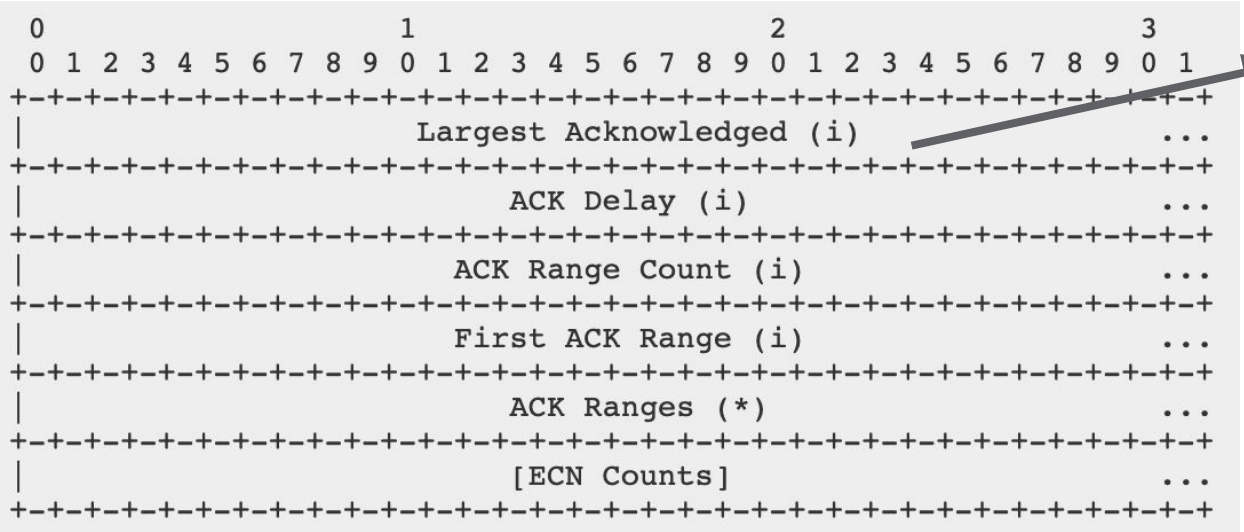
QUIC Packetization: Example



ACK Frame

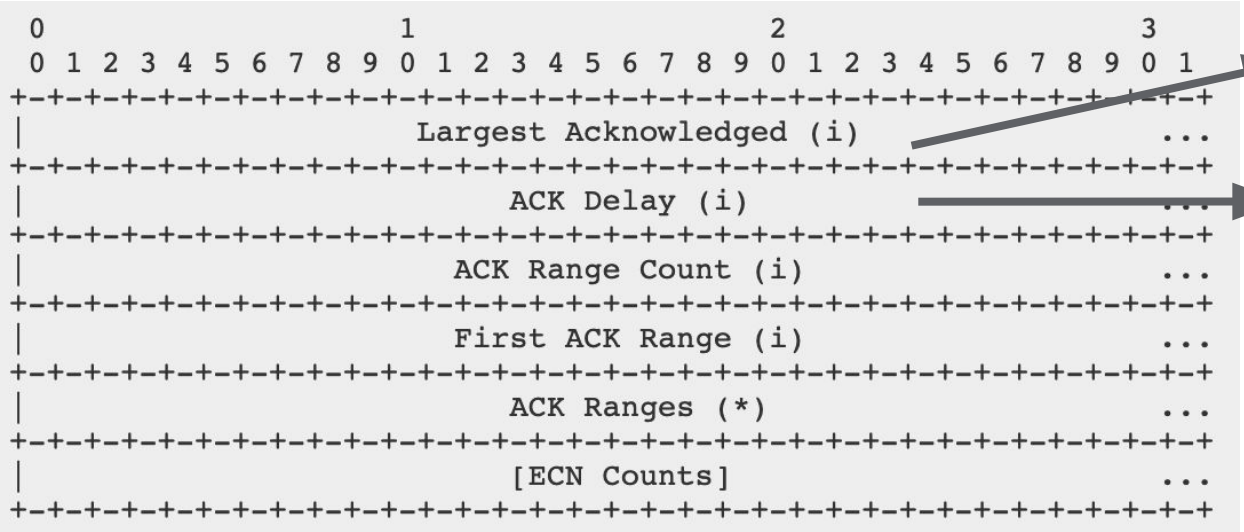

0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Largest Acknowledged (i) ...
ACK Delay (i) ...
ACK Range Count (i) ...
First ACK Range (i) ...
ACK Ranges (*) ...
[ECN Counts] ...

ACK Frame



Highest Packet Number seen so far

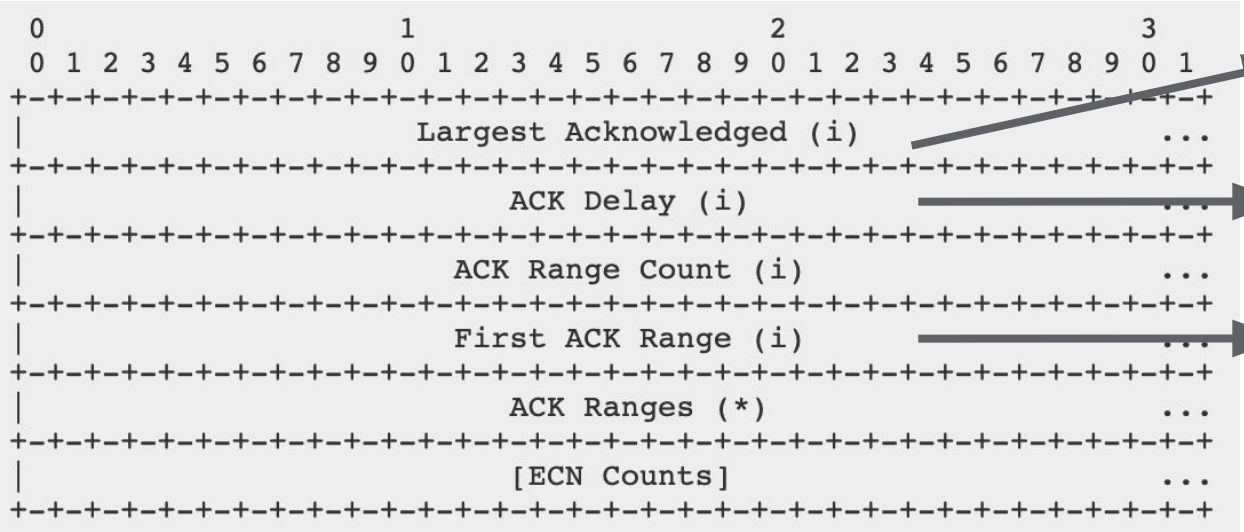
ACK Frame



Highest Packet Number seen so far

Time since Largest Acked was received

ACK Frame


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Largest Acknowledged (i) ...
ACK Delay (i) ...
ACK Range Count (i) ...
First ACK Range (i) ...
ACK Ranges (*) ...
[ECN Counts] ...
Highest Packet Number seen so far
Time since Largest Acked was received
Contiguous from Largest Acked

QUIC Packetization: Ack Example

Packets received: 1 … 125

Time since largest received: 25ms

represented as a shifted value (default 3, negotiable)

25ms = 3125us << 3

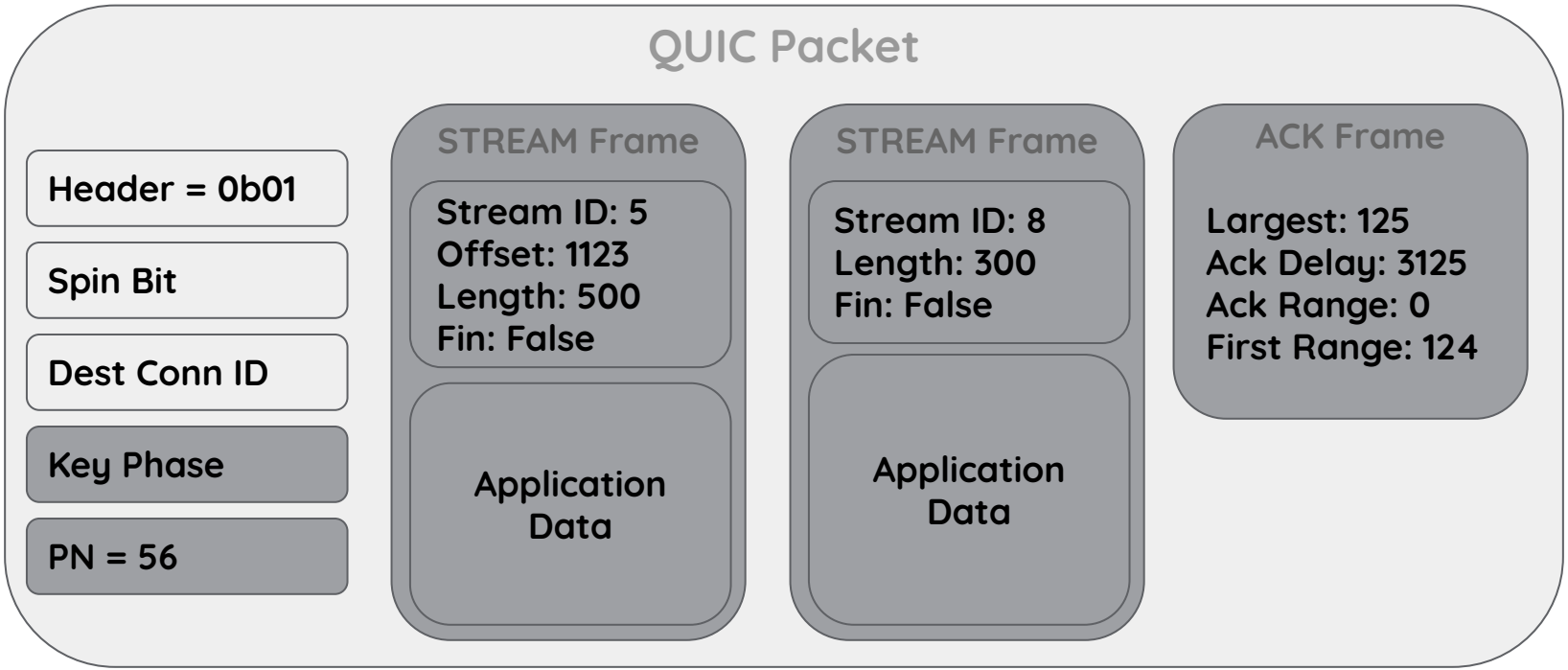
ACK fields

Largest packet received so far: 125

First Ack Range: 124

Ack Range Count: 0

QUIC Packetization: Example



QUIC Packetization: Ack Example

Packets received: 1 … 125, 130
Time since largest received: 0ms

ACK fields

Largest packet received so far: 130
First Ack Range: 0
Gap #1: 126 - 129
Ack Range #1: 1 - 125

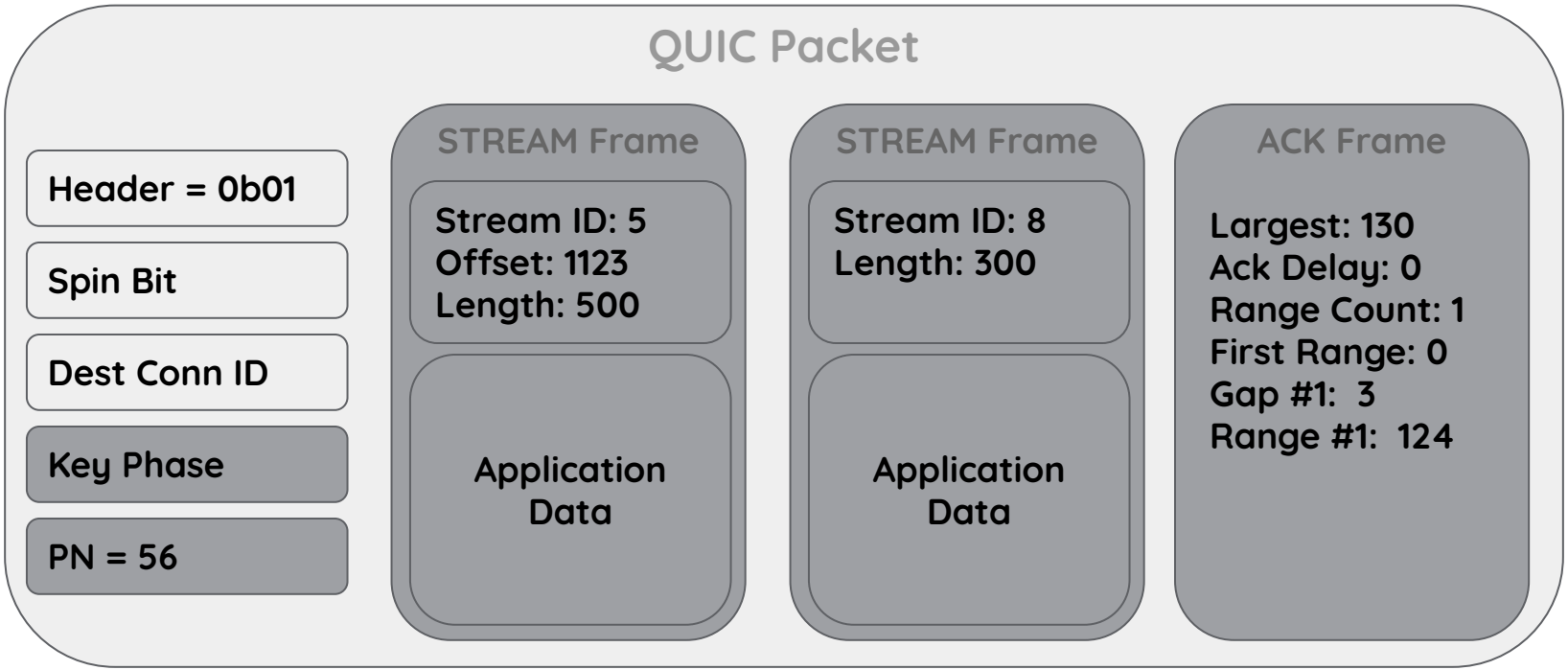
QUIC Packetization: Ack Example

Packets received: 1 … 125, 130
Time since largest received: 0ms

ACK fields

Largest packet received so far: 130
First Ack Range: 0
Gap #1: 126 - 129 (encoded as 3)
Ack Range #1: 1 - 125 (encoded as 124)

QUIC Packetization: Example



QUIC Packetization: Loss Example

Packet 56 dropped

QUIC Packetization: Loss Example

Packet 56 dropped

Also, Stream 8 was reset

QUIC Packetization: Loss Example

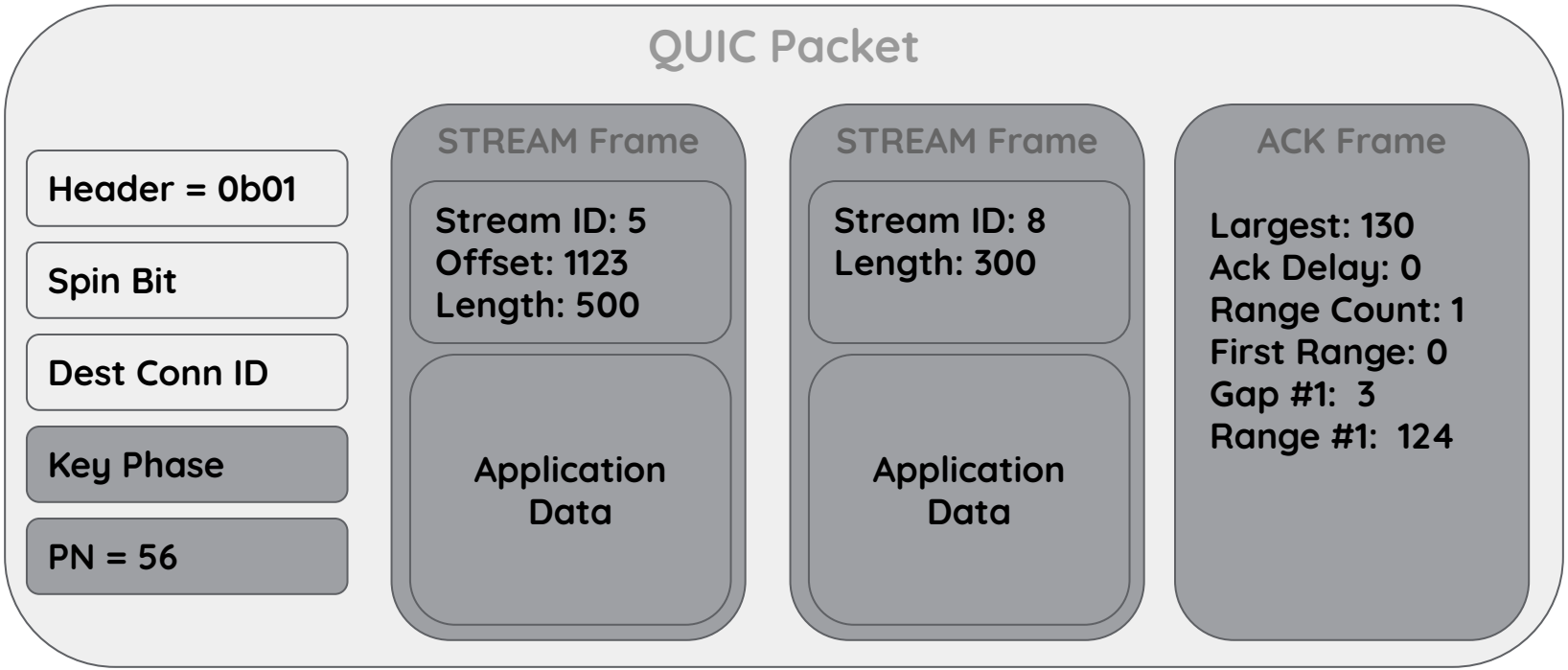
Packet 56 dropped

Also, Stream 8 was reset

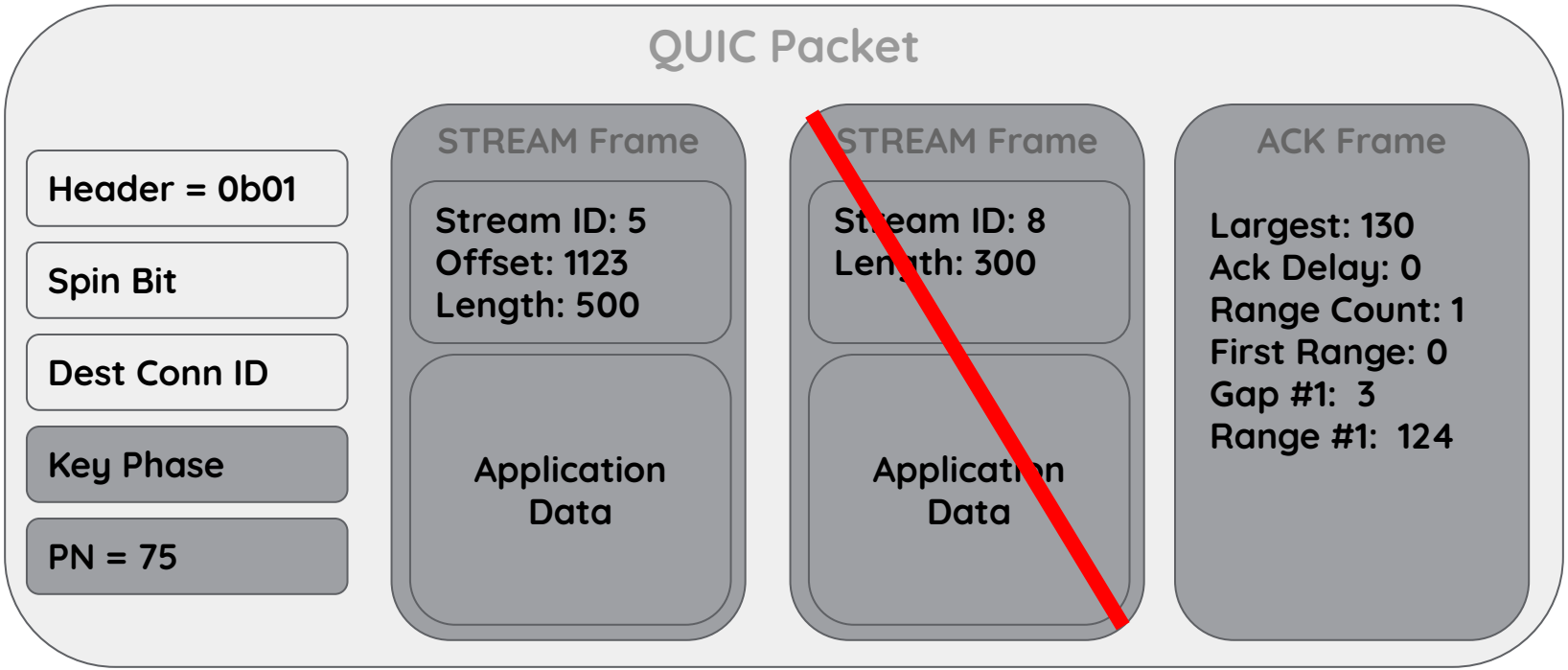
QUIC loss detection marks packet 56 as lost

let’s say last packet sent was packet number 74

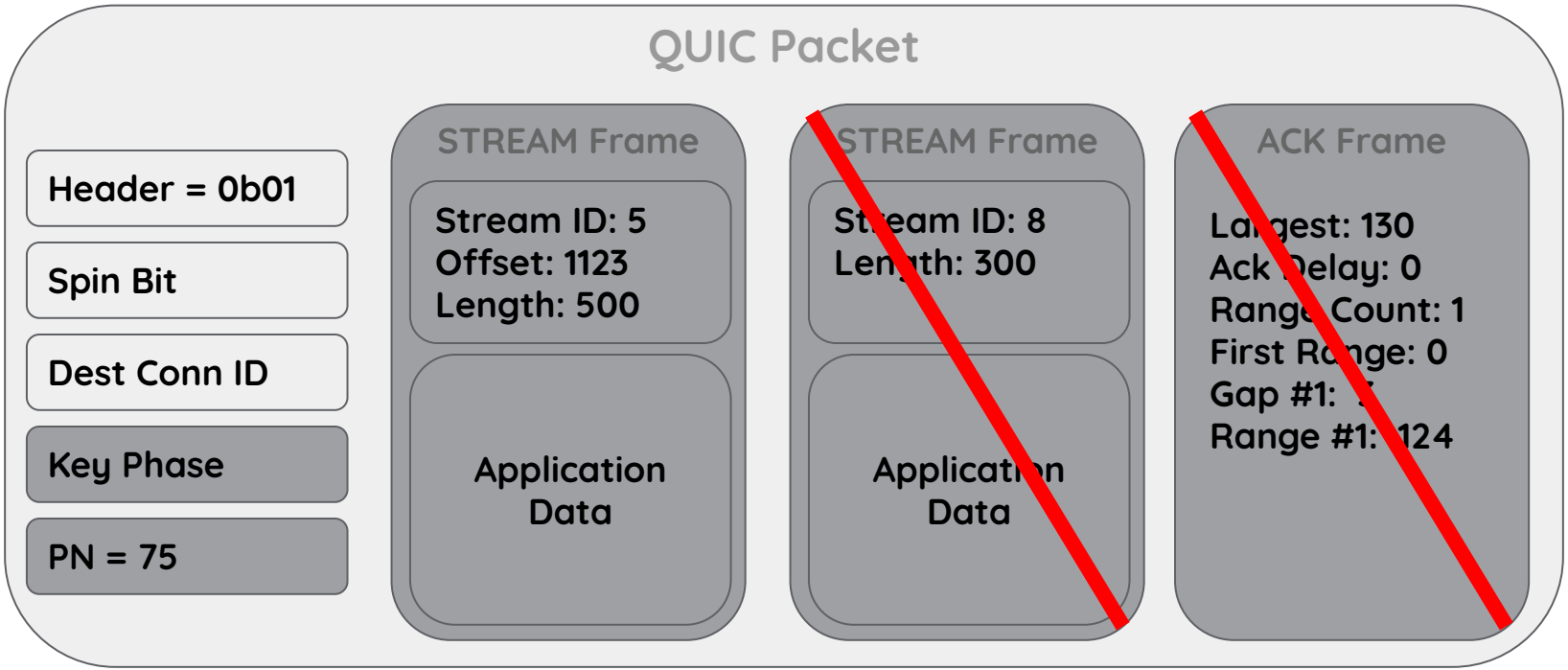
QUIC Packetization: Example



QUIC Packetization: Example



QUIC Packetization: Example



QUIC Loss Recovery

Packet numbers represent transmission order

stream IDs and offsets used for delivery order

monotonically increasing 62-bit packet numbers

(caveat: multiple PN spaces during connection setup)

QUIC Loss Recovery

Packet numbers represent transmission order

stream IDs and offsets used for delivery order

monotonically increasing 62-bit packet numbers

(caveat: multiple PN spaces during connection setup)

Packets are containers

carry a mix of various types of frames

QUIC Loss Recovery

Packet numbers represent transmission order

stream IDs and offsets used for delivery order
monotonically increasing 62-bit packet numbers
(caveat: multiple PN spaces during connection setup)

Packets are containers

carry a mix of various types of frames

Retransmissions are not automatically high priority

depends on relative stream priority
application-dependent

Generating ACK frames

SHOULD ACK every other packet

subject to 25ms delayed ack timer

Generating ACK frames

SHOULD ACK every other packet

subject to 25ms delayed ack timer

SHOULD ACK immediately if:

Received packet number != largest received + 1

CE codepoint received

Generating ACK frames

SHOULD ACK every other packet

subject to 25ms delayed ack timer

SHOULD ACK immediately if:

Received packet number != largest received + 1

CE codepoint received

MAY process more packets before ACK

allows less frequent acking

QUIC Loss Detection

Loss detection only when ACK frame received

that newly acks a packet

use both packet and time thresholds

QUIC Loss Detection

Loss detection only when ACK frame received

that newly acks a packet

use both packet and time thresholds

Packet threshold

reordering >= 3 packets

QUIC Loss Detection

Loss detection only when ACK frame received

that newly acks a packet

use both packet and time thresholds

Packet threshold

reordering >= 3 packets

Time threshold

reordering >= 1 packet AND

time > 9/8 * max(SRTT, latest_RTT)

No ACKs received: Probe Timeout

Probe Timeout (PTO) triggers packet(s) when no ACK

on PTO, send 1 or 2 probe packets (new or old data)

restarted when new ACK-eliciting packet (tail) sent

exponential backoff (pto *= 2)

pto = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay

Timeout does not necessarily mean packet loss

exception: if no data to send, mark outstanding as lost

No ACKs received: Probe Timeout

Probe Timeout (PTO) triggers packet(s) when no ACK

on PTO, send 1 or 2 probe packets (new or old data)

No ACKs received: Probe Timeout

Probe Timeout (PTO) triggers packet(s) when no ACK

on PTO, send 1 or 2 probe packets (new or old data)

restarted when new ACK-eliciting packet (tail) sent

No ACKs received: Probe Timeout

Probe Timeout (PTO) triggers packet(s) when no ACK

- on PTO, send 1 or 2 probe packets (new or old data)
- restarted when new ACK-eliciting packet (tail) sent
- exponential backoff (pto *= 2)

No ACKs received: Probe Timeout

Probe Timeout (PTO) triggers packet(s) when no ACK

- on PTO, send 1 or 2 probe packets (new or old data)
- restarted when new ACK-eliciting packet (tail) sent
- exponential backoff (pto *= 2)

pto = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay

No ACKs received: Probe Timeout

Probe Timeout (PTO) triggers packet(s) when no ACK

on PTO, send 1 or 2 probe packets (new or old data)

restarted when new ACK-eliciting packet (tail) sent

exponential backoff (pto *= 2)

pto = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay

Timeout does not necessarily mean packet loss

exception: if no data to send, mark outstanding as lost

No ACKs received: Persistent Congestion

When all packets are lost over a long-enough time

(smoothed_rtt + 4 * rttvar + max_ack_delay) *
(2 ^ kPersistentCongestionThreshold - 1)

No ACKs received: Persistent Congestion

When all packets are lost over a long-enough time

(smoothed_rtt + 4 * rttvar + max_ack_delay) *
(2 ^ kPersistentCongestionThreshold - 1)

Collapse congestion window to min_cwnd

No ACKs received: Persistent Congestion

When all packets are lost over a long-enough time

(smoothed_rtt + 4 * rttvar + max_ack_delay) *
(2 ^ kPersistentCongestionThreshold - 1)

Collapse congestion window to min_cwnd

Default kPersistentCongestionThreshold = 2

same as 2 TLPs followed by an RTO

Tooling

In-network packet tracing

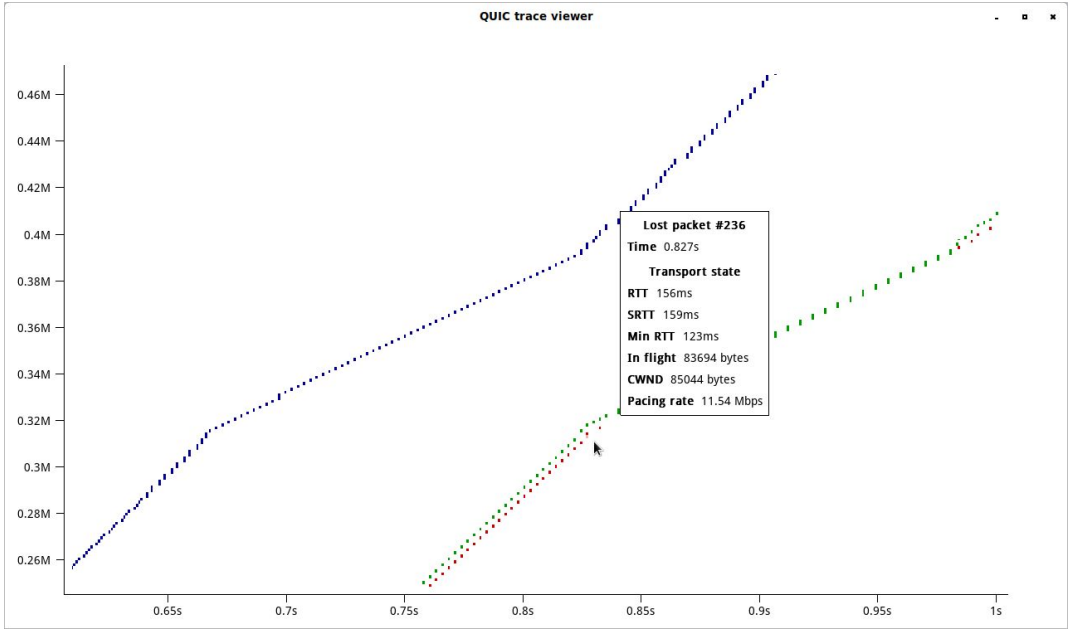
- Wireshark dissector available
- This isn't enough. Why?

Endpoint-based packet tracing

- Log packet and frame details at endpoint
- (also log other transport info, such as cwnd)
- quic-trace
- QUICvis

Tooling: quic-trace

Written by Victor Vasiliev et al (Google)
Available at https://github.com/google/quic-trace
Input: protobuf or JSON



Tooling: QUICvis

Written by Robin Marx et al
Available at https://quic.edm.uhasselt.be/
Input: JSON