QUIC Recovery & CC

Ian Swett, IETF 100

Goals

Minimize the time it takes to repair data in lost packets

Ensure the connection makes forward progress

Non-Goals

Tell implementations what to send

Principles

Packets are never retransmitted, information in them is

=> Focus on when to declare packets lost, not what to send

Wait until you have information to make a decision

=> Don’t mark packets as lost until an ack for a later packet is received

Avoid UNDO

=> F-RTO implemented by waiting for RTO verification

Notable Differences from TCP

No retransmission ambiguity

Acknowledgements contain as many ack blocks as space allows

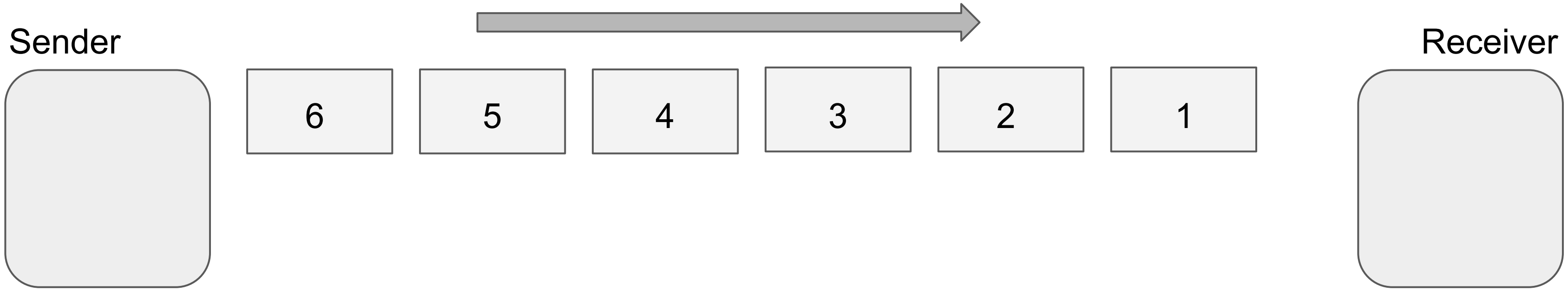
Acknowledgements are irrevocable

Acknowledgements mean a packet has been received AND processed

Recovery ends when a packet number, not a sequence number, after the largest outstanding is acknowledged

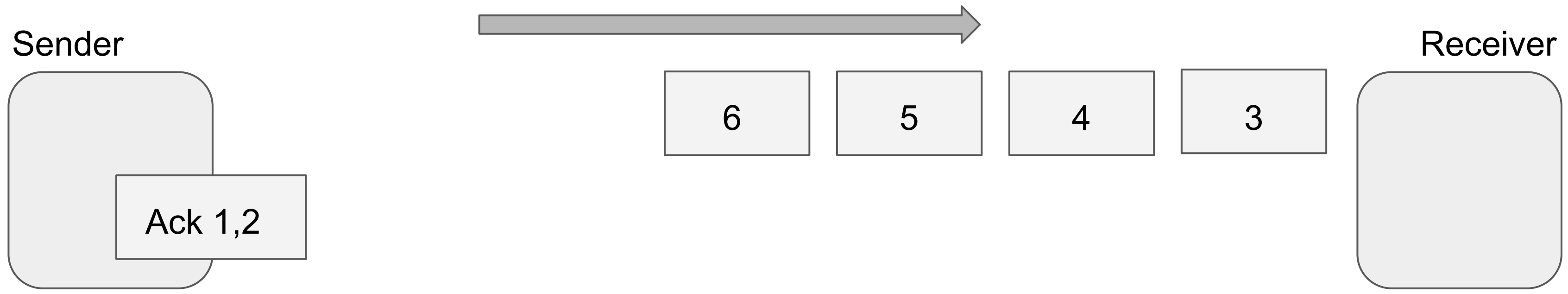
Ack Based Loss Detection: FACK style

Declare packets lost when a packet sent 3 or more later is acked



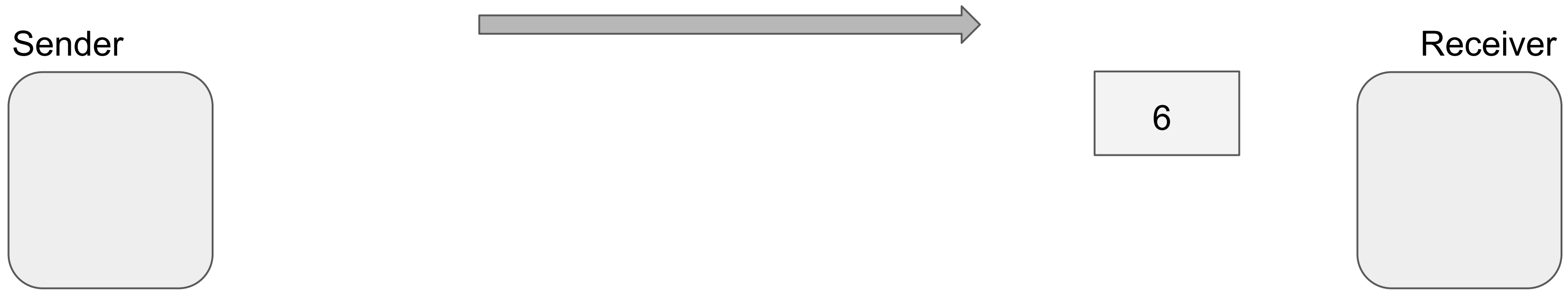
Ack Based Loss Detection: FACK style

Receive 1 and 2 and send an ACK frame



Ack Based Loss Detection: FACK style

3, 4, and 5 are dropped



Ack Based Loss Detection: FACK style

Send an ack immediately upon receiving 6

Sender
Receiver
Ack 1,2,6

Ack Based Loss Detection: FACK style

Declare 3 lost when an ack for 6 is received

- For this example, nothing is immediately retransmitted
- If we did retransmit the data in 3 as 7, it could be used to recover 4

Sender
Receiver
Ack 1,2,6
3 LOST

Ack Based Loss Detection: Early Retransmit

The largest retransmittable packet has been acked

=> Set a ⅛ RTT timer to declare other packets lost

Sender
Receiver

QUIC

Ack Based Loss Detection: Early Retransmit

Timer expires, declare 4 and 5 lost

Sender
Receiver
4 and 5 LOST

Timer Based Loss Detection

Timeout packets count towards bytes in flight

Timeout events do not cause packets to be declared lost or change congestion control

=> CWND does not change when timeouts occur

Timer Based Loss Detection: Handshake

Timeout: RTT * 2 ^ (num_handshake_timeouts + 1)

Set From: Last sent handshake frame

Send: All unacked handshake packets

RTT is smoothed RTT, or the expected RTT

Expected RTT may be based on past connections to the same host or connections to other hosts.

Timer Based Loss Detection: TLP

Timeout: Max(2*SRTT, 1.5*SRTT+kDelayedAckTimeout)

Set From: Last sent packet with retransmittable frames

Send: One packet with retransmittable frames

Timer Based Loss Detection: RTO

Timeout: Max(SRTT + 4*RTTVAR, minRTO) * 2 ^ (num_rtos)

Set From: Last sent packet with retransmittable frames

Send: Two packets with retransmittable frames

If RTO is not spurious:

- CWND is reduced to MinCWND
- All packets prior to the acknowledged packet are lost

QUIC

Magic Numbers

Constants based on TCP RFCs and best practices

Most defaults are identical to TCP defaults in Linux

Notable Exceptions:

- Default ack delay is 25ms - Should be changeable (#912)
- Default RTT is 100ms => 200ms handshake timeout(#752)

Implementation Tips

Spurious retransmissions happen

- Continue to track data sent in lost packets for a while after they are lost

Never retransmit an ACK frame, always send up to date ones

Always send packets in packet number order or your implementation will be much more complex