Connection Termination
QUIC Interim, Seattle
4 ways to terminate a connection

Stateless Reset
Immediate Close
Application Close
Idle Timeout
**Stateless Reset**

Only used when server has no connection state at all

Server sends with Stateless Reset Token

Client tears down all state immediately
Immediate Close

Error close
used for app or transport error cases
or for an app that knows what it’s doing (NO_ERROR) ?

On Immediate Close
All streams considered implicitly reset
CONNECTION_CLOSE frame sent
Connection enters “draining” period

During draining period
Respond with CONNECTION_CLOSE
Exit if CONNECTION_CLOSE received
Exit if ACK for CONNECTION_CLOSE received
Application Close

(Some of this is in draft, details need discussion)

Application peers negotiate termination
   Hypothetical example: by closing stream 1 for HTTP

Connection is in CLOSE_PENDING state (not in spec)
   No new data should be received in this state
   No new data should be queued by app
   Transport finishes sending pending data and rtxs

When nothing left to send, enter “draining” period
   Send same final ACK in response to received packets
Draining Period

After sending last packet, wait to handle retransmissions

Last packet may be lost in the network
Need to respond to peer’s retransmissions

How long? Currently 3 x RTO
Allows 2 RTOs from peer
(Peer’s RTO may be different than ours)

ACK, CONNECTION_CLOSE, PADDING allowed to be sent

After draining period, discard connection state
Idle Timeout

(Basic text in draft, details need specification)

Connection is *idle* for longer than idle timeout
  Timeout negotiated during handshake

*_Idle* is defined as time from when: (no spec text)
  Last new app data sent or received
  Last PING sent, retransmissions excluded
  Last PING received

What if path is dead?
  Retransmissions being sent, no acks received
  *Idle* is app semantic, need something when path dies
Path Timeout (not in spec)

Duration after which path is declared dead
(i) time since first unacked packet
(ii) this packet must be one for which ack is expected
(iii) packet number > largest acked so far (why?)
Should be f(RTO) (with some max)
Locally configurable
Path Timeout (not in spec)

Duration after which path is declared dead
(i) time since first unacked packet
(ii) this packet must be one for which ack is expected
(iii) packet number > largest acked so far (why?)
Should be $f(RTO)$ (with some max)
Locally configurable

Path Timeout Haiku

Time since endpoint sent
the first ackable packet
after largest acked