Transport Layer
Part 6

Mark Allman
mallman@case.edu

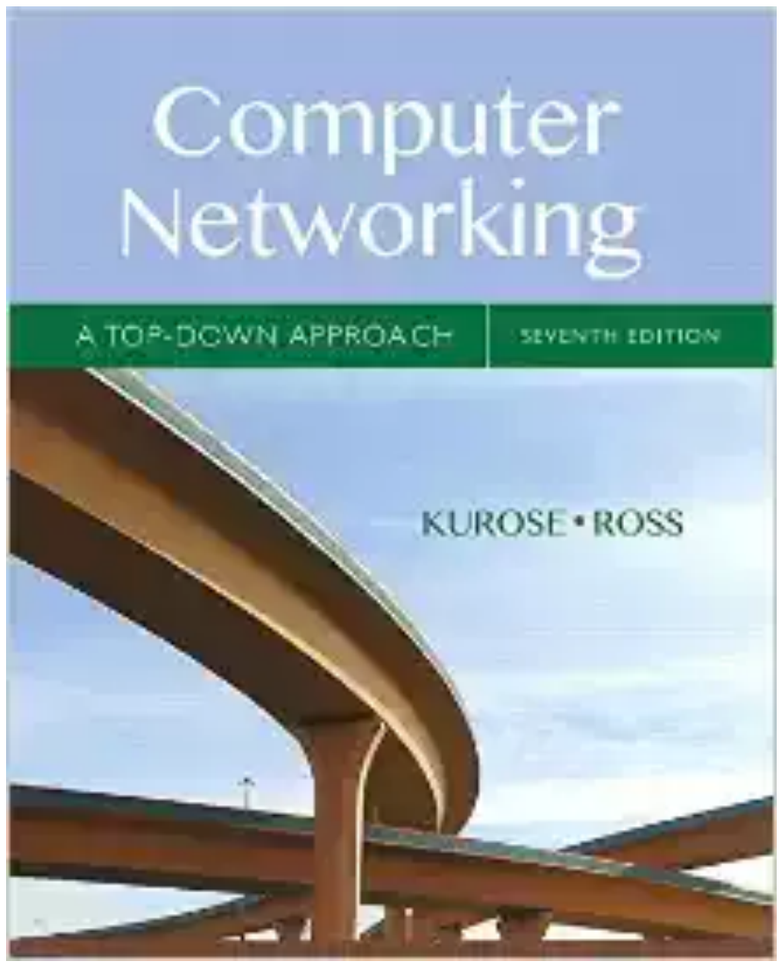
EECS 325/425
Fall 2018

"We're the junk yard dogs, we're the alley cats,
Keep the wind at our front and the hell at our back"

These slides are more-or-less directly from the slide set developed by Jim Kurose and Keith Ross for their book "Computer Networking: A Top Down Approach, 5th edition".

The slides have been lightly adapted for Mark Allman's EECS 325/425 Computer Networks class at Case Western Reserve University.

Reading Along ...



- 3.5: Connection-oriented transport: TCP
  - reliable data transfer

TCP reliable data transfer

- TCP creates rdt service on top of IP's unreliable service
- pipelined segments
- cumulative acks
- TCP uses single retransmission timer
- retransmissions are triggered by:
  - timeout events
  - duplicate acks
- initially consider simplified TCP sender:
  - ignore flow control, congestion control

TCP sender events:

TCP sender events:

data rcvd from app:

- Create segment with seq #
- seq # is byte-stream number of first data byte in segment
- start timer if not already running (think of timer as for oldest unacked segment)
- expiration interval: RTO

TCP sender events:

data rcvd from app:

- Create segment with seq #
- seq # is byte-stream number of first data byte in segment
- start timer if not already running (think of timer as for oldest unacked segment)
- expiration interval: RTO

timeout:

- retransmit segment that caused timeout
- restart timer

TCP sender events:

data rcvd from app:

- Create segment with seq #
- seq # is byte-stream number of first data byte in segment
- start timer if not already running (think of timer as for oldest unacked segment)
- expiration interval: RTO

timeout:

- retransmit segment that caused timeout
- restart timer

Ack rcvd:

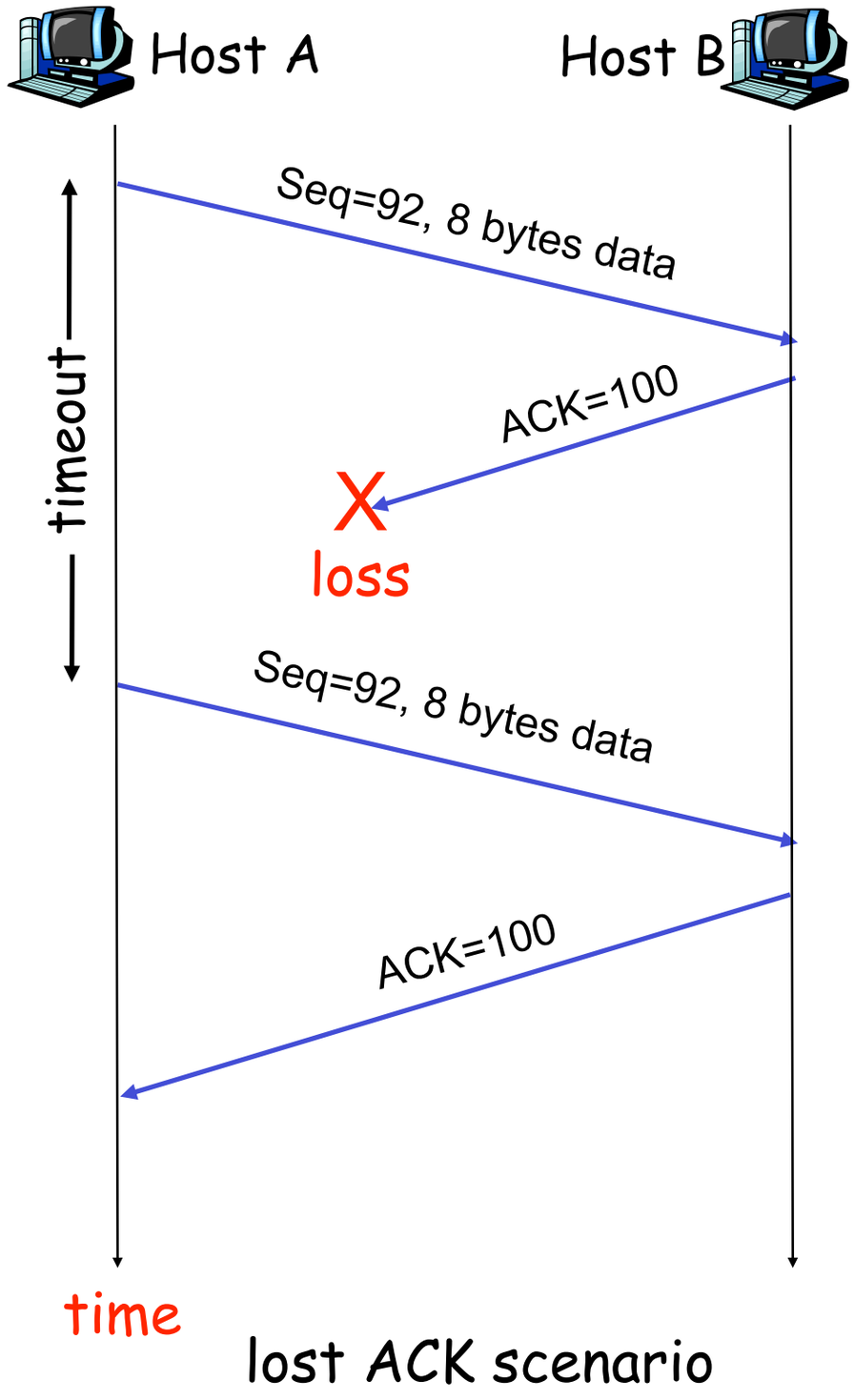
- If acknowledges previously unacked segments
  - update what is known to be acked
  - start timer if there are outstanding segments

TCP sender (simplified)

```
NextSeqNum = InitialSeqNum
SendBase = InitialSeqNum

loop (forever) {
   switch(event)

   event: data received from application above
        create TCP segment with sequence number NextSeqNum
        if (timer currently not running)
             start timer
        pass segment to IP
        NextSeqNum = NextSeqNum + length(data)

   event: timer timeout
        retransmit not-yet-acknowledged segment with
             smallest sequence number
        start timer

   event: ACK received, with ACK field value of y
        if (y > SendBase) {
             SendBase = y
             if (there are currently not-yet-acknowledged segments)
                  start timer
             }

} /* end of loop forever */
```

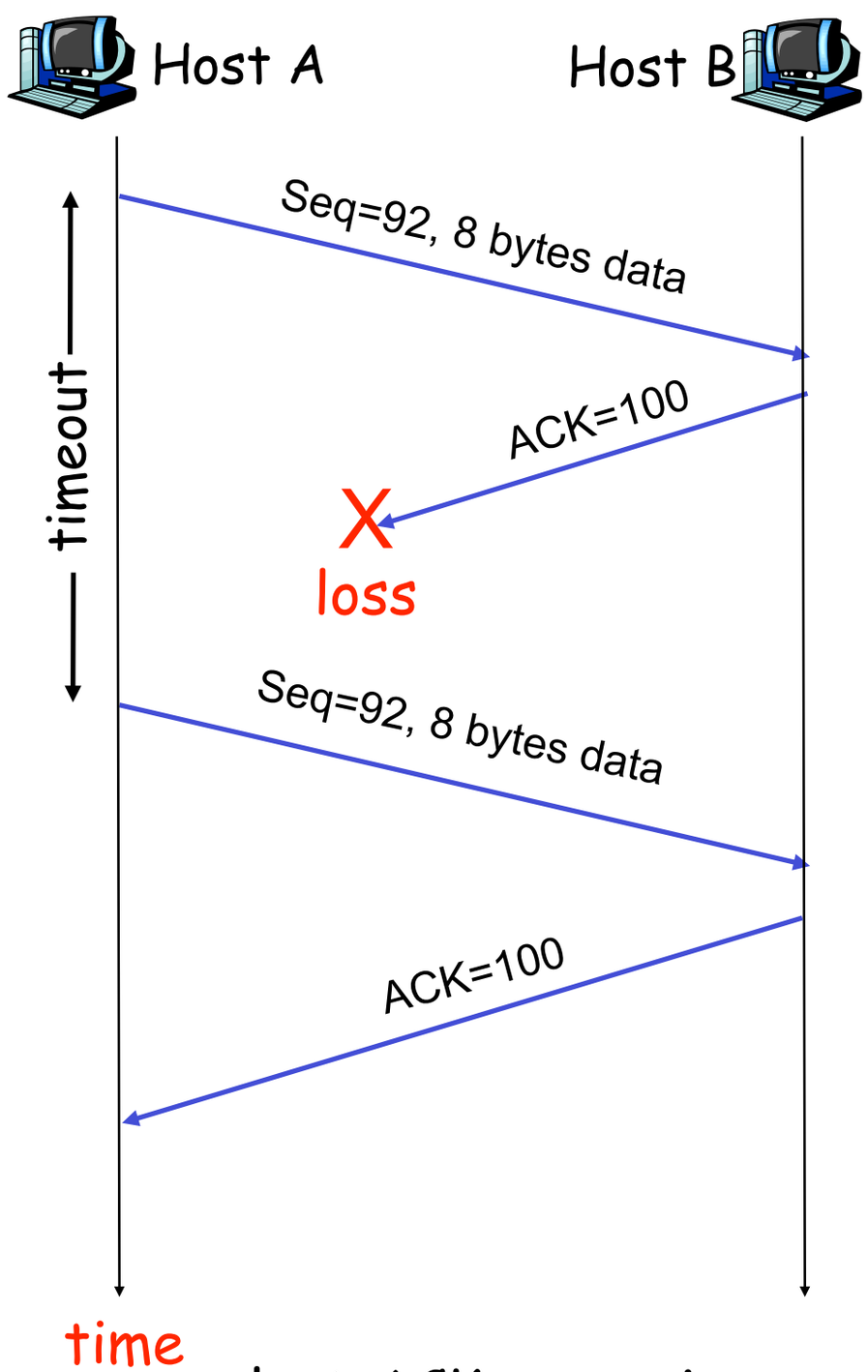
TCP: retransmission scenarios

TCP: retransmission scenarios

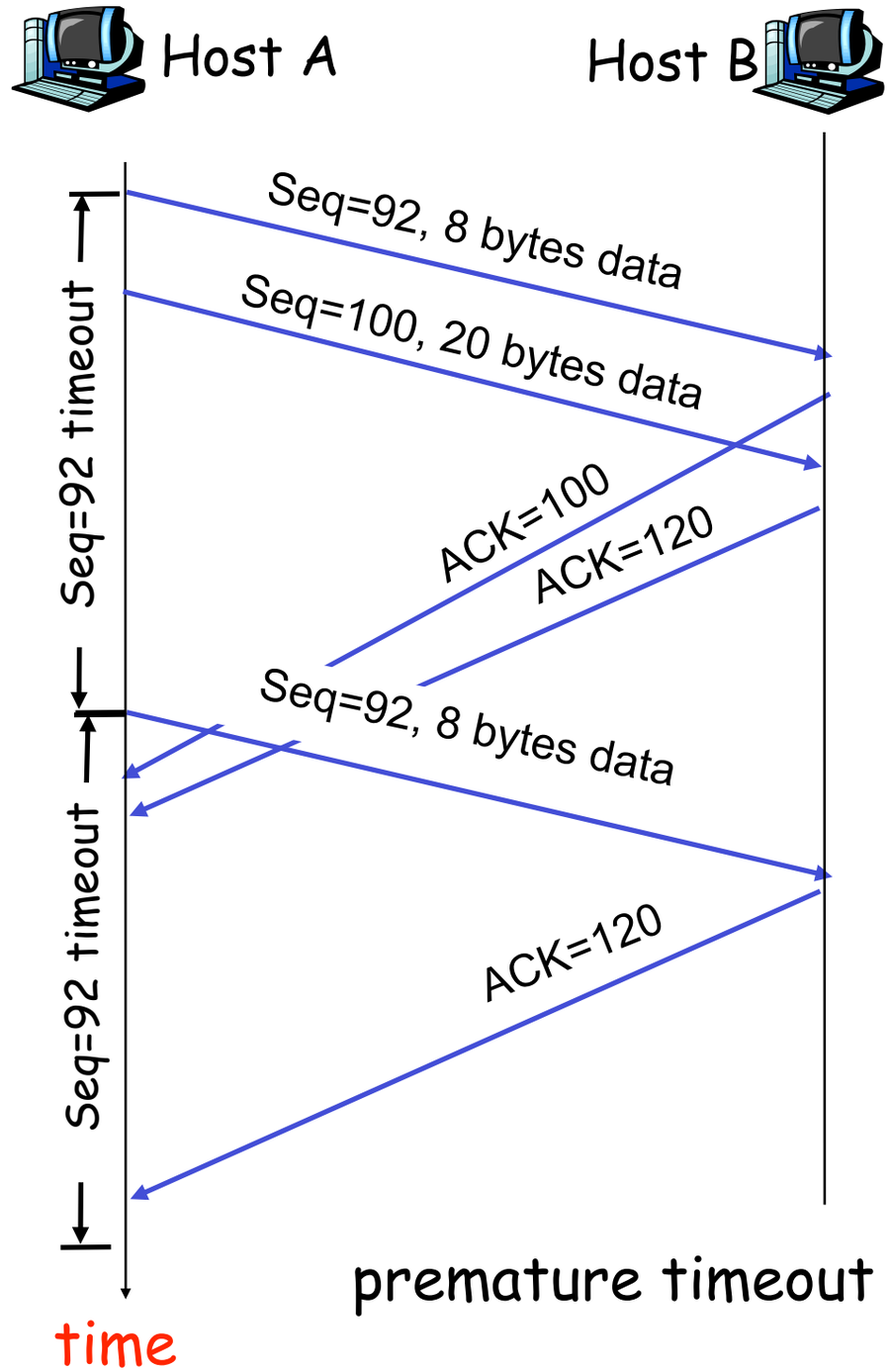


lost ACK scenario

TCP: retransmission scenarios

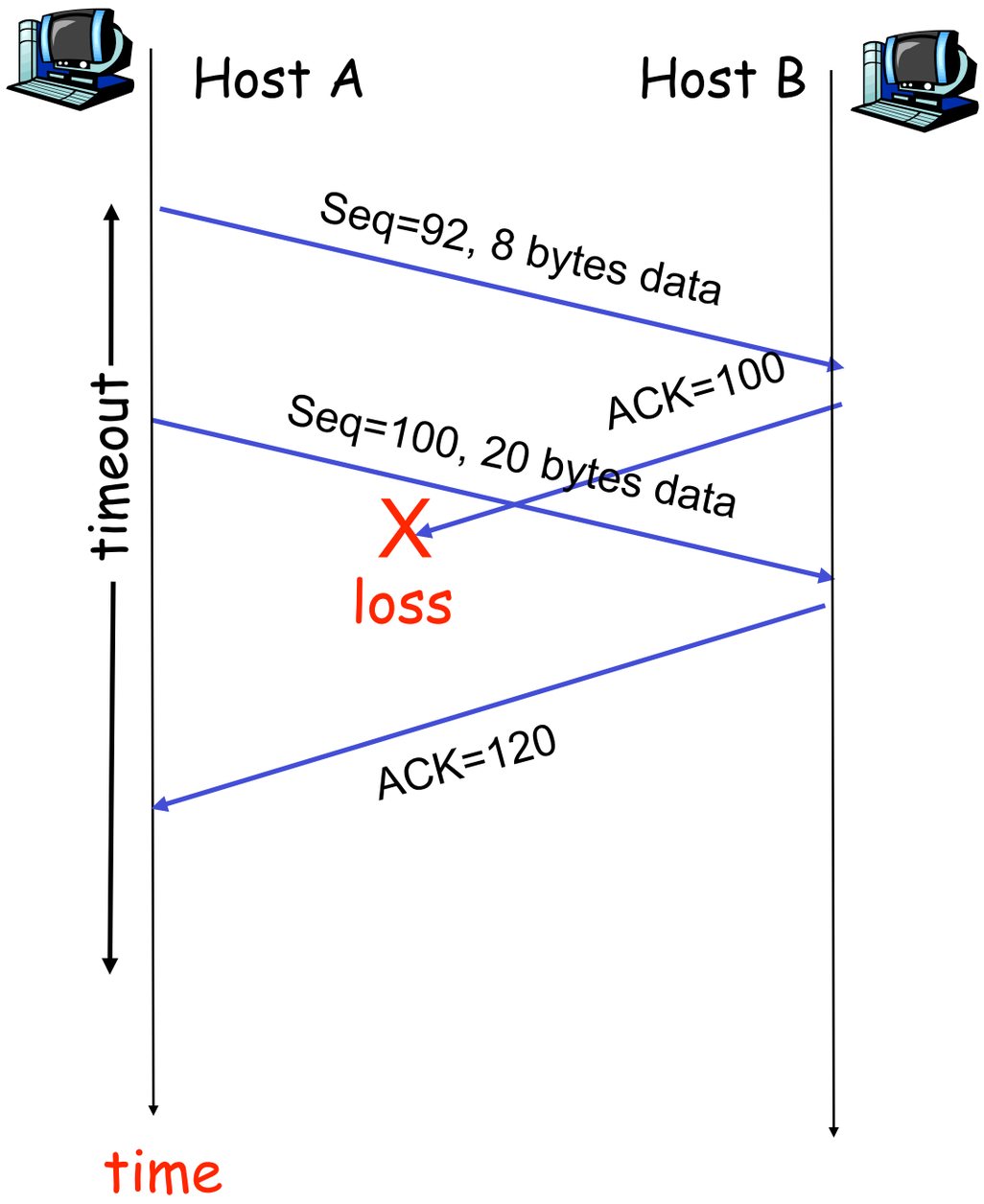


lost ACK scenario



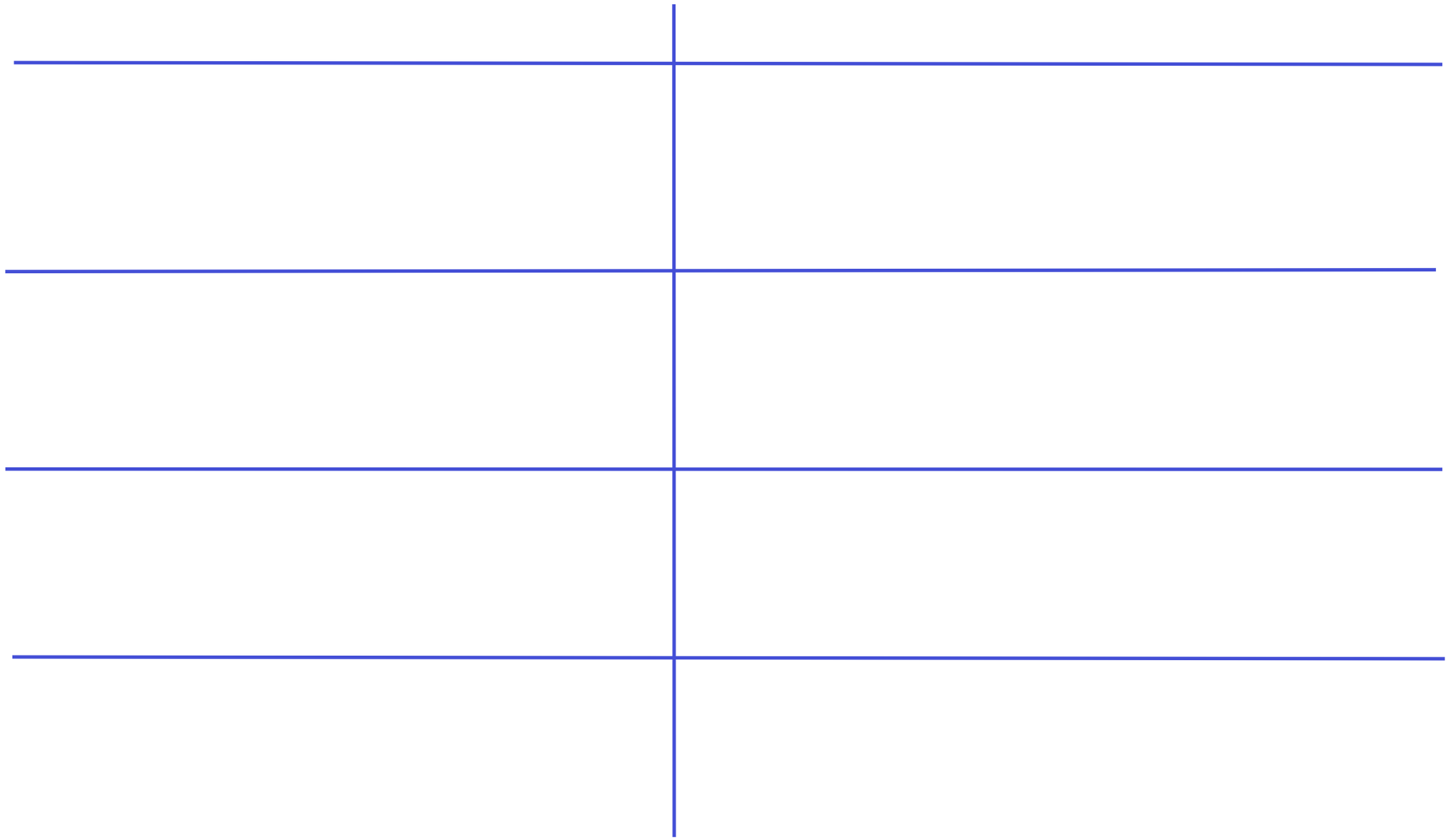
premature timeout

TCP retransmission scenarios (more)



Cumulative ACK scenario

TCP ACK generation [RFC 1122, RFC 5681]



TCP ACK generation [RFC 1122, RFC 5681]

Event at Receiver	TCP Receiver action

TCP ACK generation [RFC 1122, RFC 5681]

Event at Receiver	TCP Receiver action
Arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	

TCP ACK generation [RFC 1122, RFC 5681]

Event at Receiver	TCP Receiver action
Arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	Delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK

TCP ACK generation [RFC 1122, RFC 5681]

Event at Receiver	TCP Receiver action
Arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	Delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
Arrival of in-order segment with expected seq #. One other segment has ACK pending	

TCP ACK generation [RFC 1122, RFC 5681]

Event at Receiver	TCP Receiver action
Arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	Delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
Arrival of in-order segment with expected seq #. One other segment has ACK pending	Immediately send single cumulative ACK, ACKing both in-order segments

TCP ACK generation [RFC 1122, RFC 5681]

Event at Receiver	TCP Receiver action
Arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	Delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
Arrival of in-order segment with expected seq #. One other segment has ACK pending	Immediately send single cumulative ACK, ACKing both in-order segments
Arrival of out-of-order segment higher-than-expect seq. # . Gap detected	

TCP ACK generation [RFC 1122, RFC 5681]

Event at Receiver	TCP Receiver action
Arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	Delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
Arrival of in-order segment with expected seq #. One other segment has ACK pending	Immediately send single cumulative ACK, ACKing both in-order segments
Arrival of out-of-order segment higher-than-expect seq. # . Gap detected	Immediately send *duplicate ACK*, indicating seq. # of next expected byte

TCP ACK generation [RFC 1122, RFC 5681]

Event at Receiver	TCP Receiver action
Arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	Delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
Arrival of in-order segment with expected seq #. One other segment has ACK pending	Immediately send single cumulative ACK, ACKing both in-order segments
Arrival of out-of-order segment higher-than-expect seq. # . Gap detected	Immediately send *duplicate ACK*, indicating seq. # of next expected byte
Arrival of segment that partially or completely fills gap	

TCP ACK generation [RFC 1122, RFC 5681]

Event at Receiver	TCP Receiver action
Arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	Delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
Arrival of in-order segment with expected seq #. One other segment has ACK pending	Immediately send single cumulative ACK, ACKing both in-order segments
Arrival of out-of-order segment higher-than-expect seq. # . Gap detected	Immediately send *duplicate ACK*, indicating seq. # of next expected byte
Arrival of segment that partially or completely fills gap	Immediate send ACK, provided that segment starts at lower end of gap