



Transport Layer – Time Out, Connection Management

Lec 16

TCP reliable data transfer

- TCP creates rdt service on top of IP's unreliable service
 - pipelined segments
 - cumulative acks
 - single retransmission timer
- retransmissions triggered by:
 - timeout events
 - duplicate acks

let's initially consider
simplified TCP sender:

- ignore duplicate acks
- ignore flow control, congestion control

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: **TimeoutInterval**

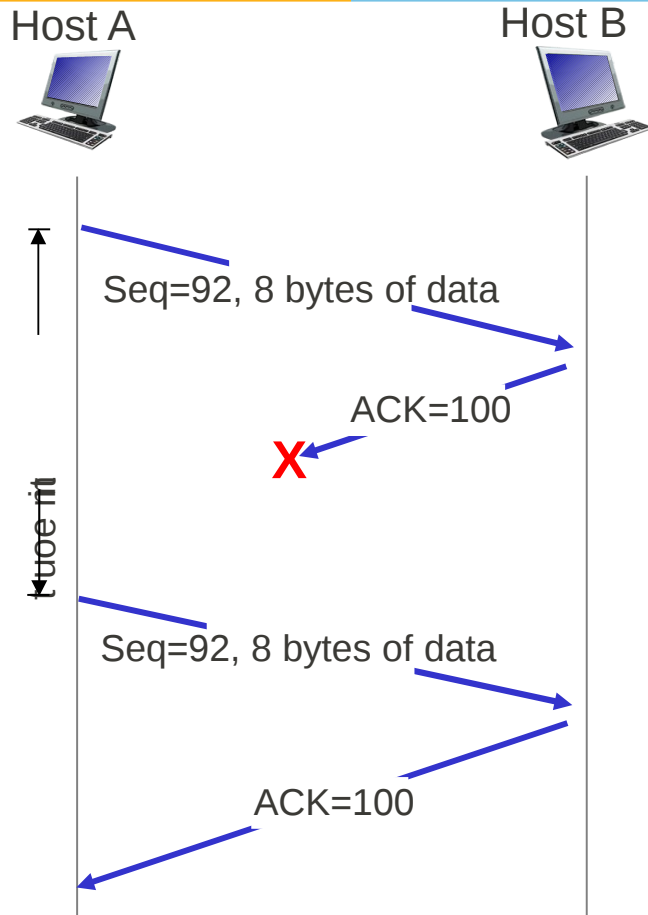
timeout:

- retransmit segment that caused timeout
- restart timer

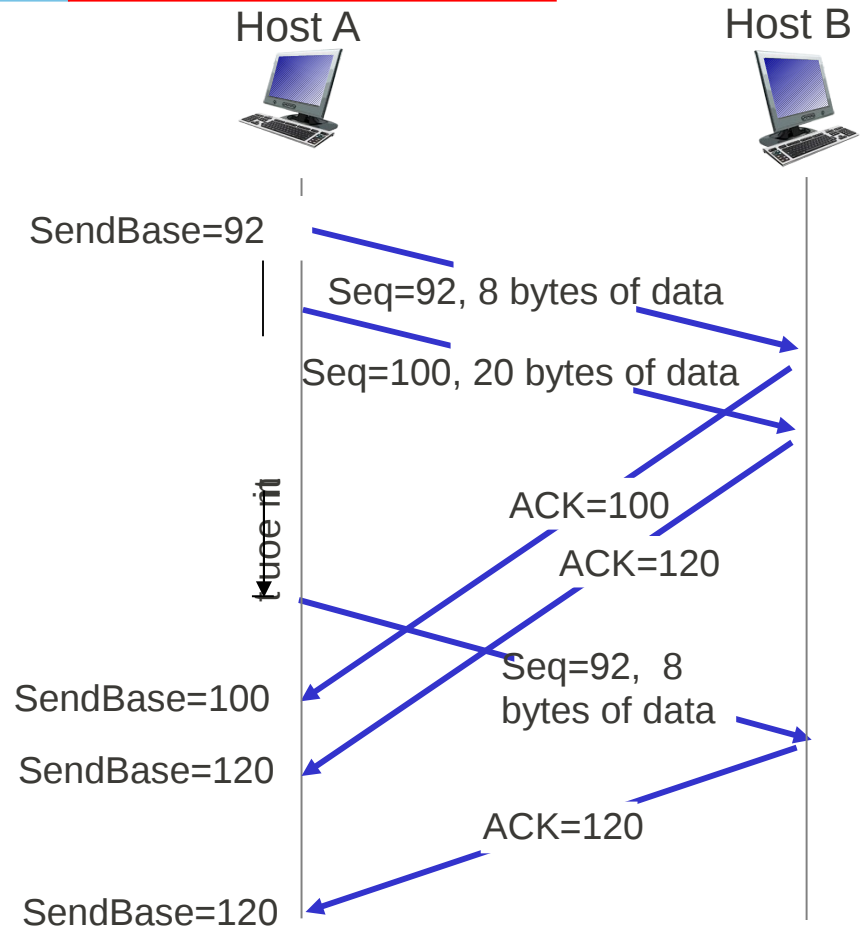
ack rcvd:

- if ack acknowledges previously unacked segments
 - update what is known to be ACKed
 - start timer if there are still unacked segments

TCP: retransmission scenarios

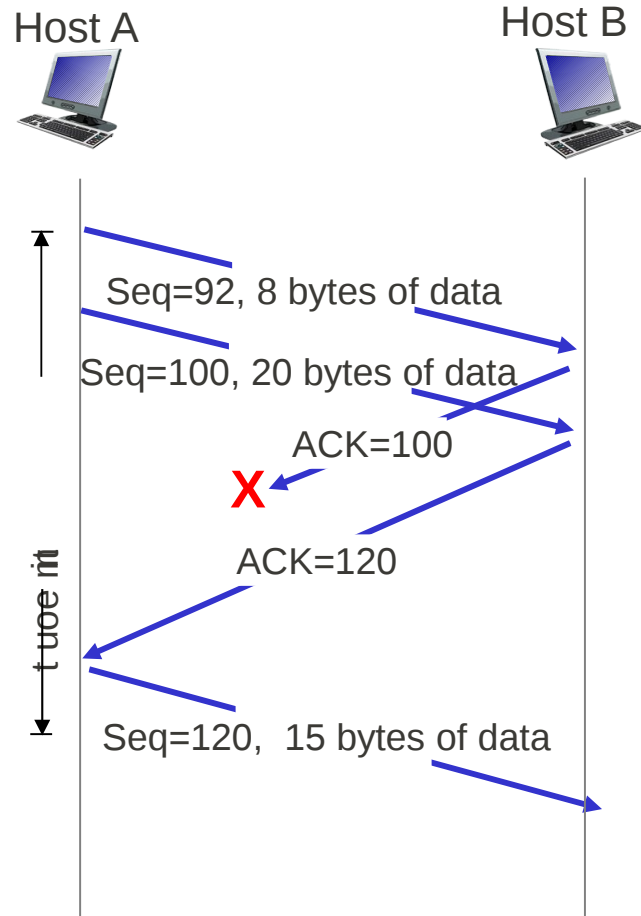


lost ACK scenario



premature timeout

TCP: retransmission scenarios



cumulative ACK

TCP ACK generation



[RFC 1122, RFC 2581]

<i>event at receiver</i>	<i>TCP receiver action</i>
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 <i>duplicate ACK</i> , 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

TCP fast retransmit

- time-out period often relatively long:
 - long delay before resending lost packet
- detect lost segments via duplicate ACKs.
 - sender often sends many segments back-to-back
 - if segment is lost, there will likely be many duplicate ACKs.

TCP fast retransmit

if sender receives 3 ACKs for same data (“triple duplicate ACKs”), resend unacked segment with smallest seq #

- likely that unacked segment lost, so don’t wait for timeout

(“triple duplicate ACKs”),

The diagram illustrates a Stop-and-Wait protocol scenario between Host A and Host B. The sequence of events is as follows:

- Host A sends Seq=92, 8 bytes of data.
- Host B receives Seq=92, 8 bytes of data.
- Host A sends Seq=100, 20 bytes of data.
- The packet Seq=100, 20 bytes of data is lost, indicated by a red 'X'.
- Host B receives Seq=92, 8 bytes of data (retransmitted).
- Host B sends ACK=100.
- Host A receives ACK=100.
- Host A sends ACK=100.
- Host B receives ACK=100.
- Host A sends ACK=100.
- Host B receives ACK=100.
- Host A sends Seq=100, 20 bytes of data.

A vertical double-headed arrow on the left side of the diagram is labeled "timeout", indicating the period during which Host A waits for an acknowledgment before retransmitting the data.

fast retransmit after sender receipt of triple duplicate ACK