CIS 457 - Data Communications

Nathan Bowman

Images taken from Kurose and Ross book

Reliable Data Transfer in TCP

TCP has mechanisms discussed previously to help with reliable data transfer

- checksums (like UDP)
- sequence numbers
- ACKs
- timeouts

TCP uses these mechanisms in slightly different way than either GBN or SR protocols, but similar

TCP is based on sliding window idea

However, sliding window in TCP is not fixed size

Size based on available space at receiver -- will discuss more when describing flow control

Like GBN, TCP uses just one timer at once (for oldest un-ACK'd packet)

Unlike GBN, when timer goes off, TCP resends only one packet

As mentioned, most TCP receivers will buffer out-of-order packets

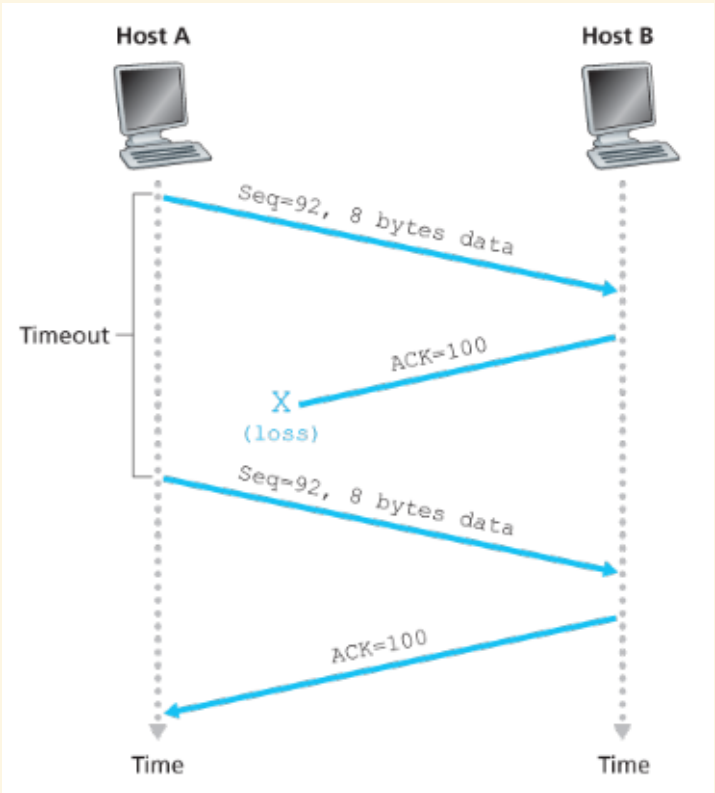
In examples, concern ourselves with data transfer in just one direction

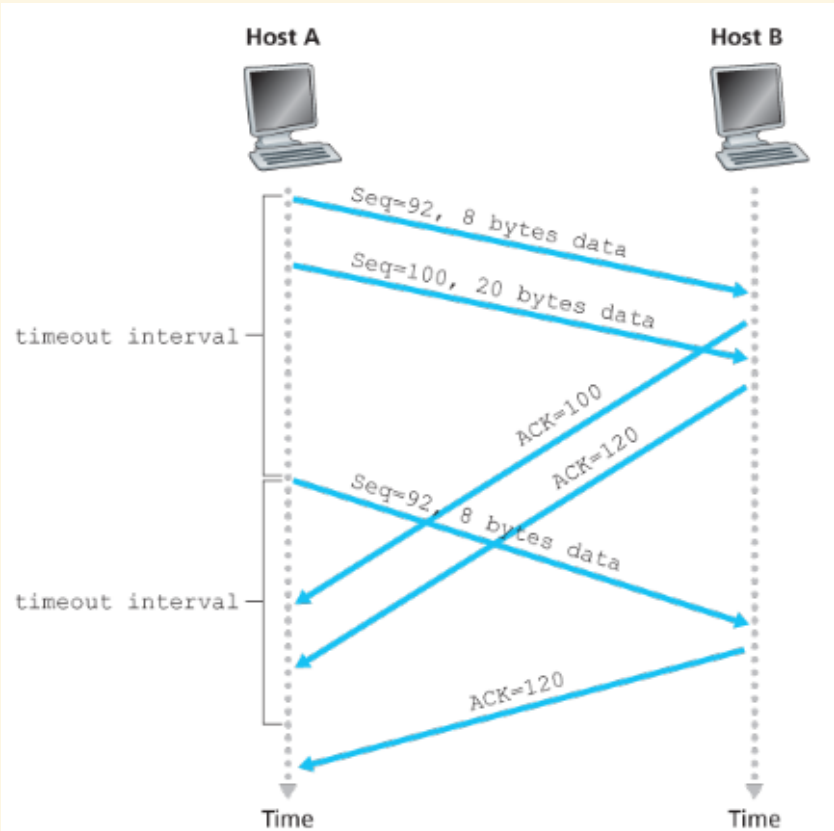
```
/* Assume sender is not constrained by TCP flow or congestion control, that data from above is less
than MSS in size, and that data transfer is in one direction only. */

NextSeqNum=InitialSeqNumber
SendBase=InitialSeqNumber

loop (forever) {
    switch(event)
```

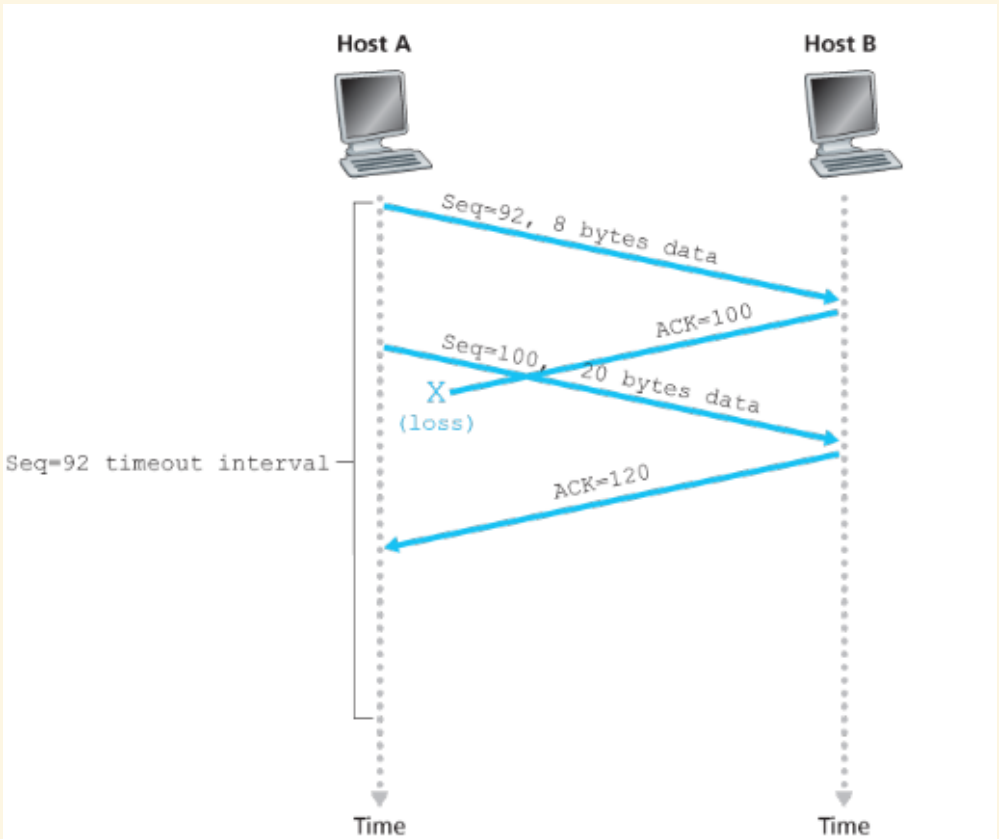
```
event: data received from application above
    create TCP segment with sequence number NextSeqNum
    if (timer currently not running)
        start timer
    pass segment to IP
    NextSeqNum=NextSeqNum+length(data)
    break;
```

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

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


Host A
Host B
Seq=92, 8 bytes data
Timeout
ACK=100
X
(loss)
Seq=92, 8 bytes data
ACK=100
Time
Time


Host A
Host B
Seq=92, 8 bytes data
Seq=100, 20 bytes data
timeout interval
ACK=100
ACK=120
Seq=92, 8 bytes data
timeout interval
ACK=120
Time
Time


Host A
Host B
Seq=92, 8 bytes data
ACK=100
Seq=100, 20 bytes data
X
(loss)
Seq=92 timeout interval
ACK=120
Time
Time

There are a few tweaks generally made to protocols

Generally, sample RTT used to compute expected time and set timeout according to formula from previous lecture

However, after timeout occurs, packet is re-sent and timeout doubled

Each time same packet times out, timeout is doubled again (growing exponentially)

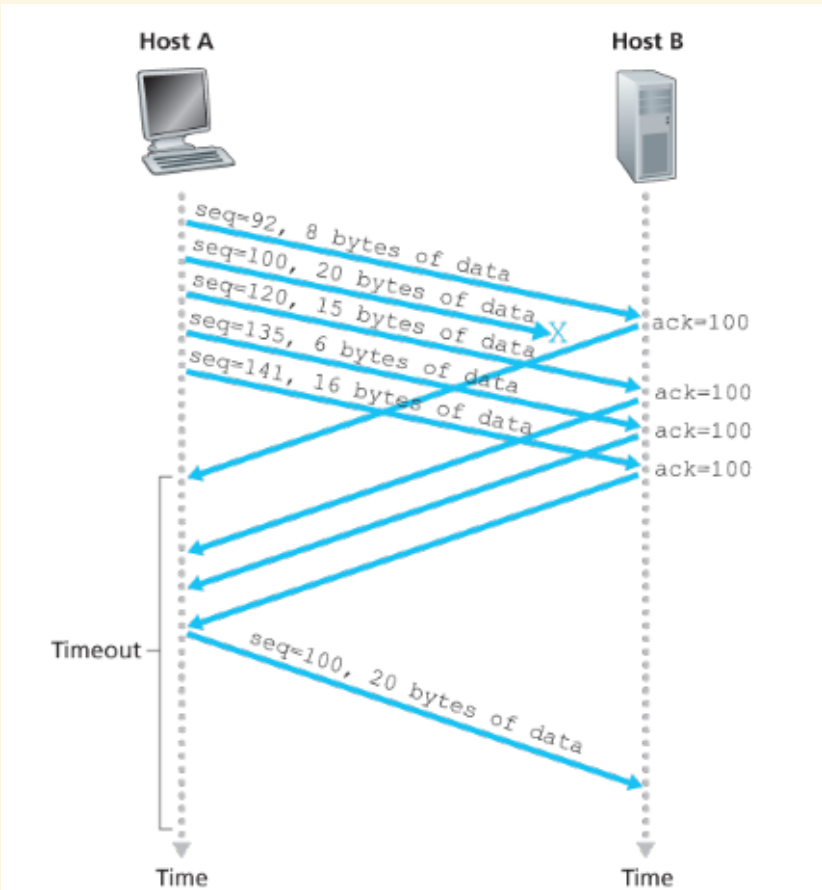
Dropped packets often caused by congestion, so increasing timeouts helpful to network as a whole by reducing congestion

After packet successfully ACK'd, go back to previous formula for timeout

If confident packet has been lost, waiting until timeout to resend is wasteful

When TCP sender receives three consecutive duplicate ACKs for same byte, indicates packet was likely lost and subsequent packets are being received

Triggers **fast retransmit** -- sender re-sends packet immediately, without waiting for timeout


Host A
Host B
seq=92, 8 bytes of data
seq=100, 20 bytes of data
seq=120, 15 bytes of data
seq=135, 6 bytes of data
seq=141, 16 bytes of data
X
ack=100
ack=100
ack=100
ack=100
Timeout
seq=100, 20 bytes of data
Time
Time

Receiver Responses to Events

Arrival of in-order segment with expected sequence number. All data up to expected sequence number already acknowledged.	Delayed ACK. Wait up to 500 msec for arrival of another in-order segment. If next in-order segment does not arrive in this interval, send an ACK.
Arrival of in-order segment with expected sequence number. One other in-order segment waiting for ACK transmission.	One Immediately send single cumulative ACK, ACKing both in-order segments.
Arrival of out-of-order segment with higher-than-expected sequence number. Gap detected.	Immediately send duplicate ACK, indicating sequence number of next expected byte (which is the lower end of the gap).
Arrival of segment that partially or completely fills in gap in received data.	Immediately send ACK, provided that segment starts at the lower end of gap.