

TCP

- Adaptive Retransmission
- Flow Control
- Congestion Control
- Reliable connection startup and shutdown

Adaptive Retransmission

TCP achieves reliability by using two mechanisms:

1. Acknowledgments

- the receiver sends an acknowledgment when the data arrives.
- Note: There are two kinds of acknowledgments:
 - positive acknowledgment - send acknowledgment when data arrives
 - negative acknowledgment - send an acknowledgment when data

does not arrive

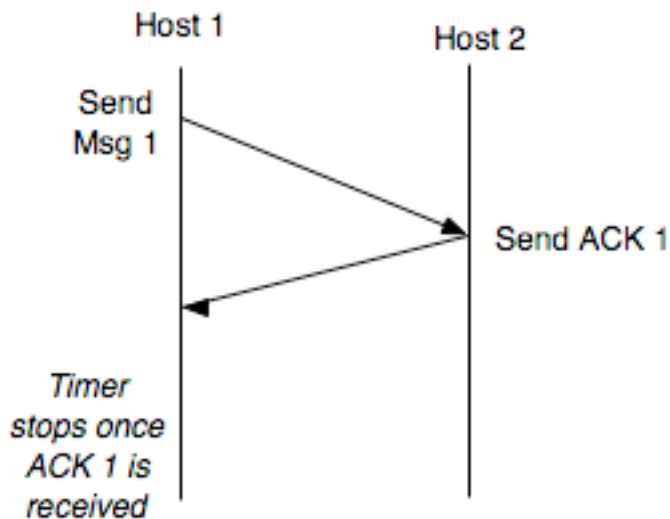
- TCP uses POSITIVE ACKNOWLEDGMENT. Few protocols use negative acknowledgment.

It could be used when data is continuously send at a fixed rate and the receiver acknowledges

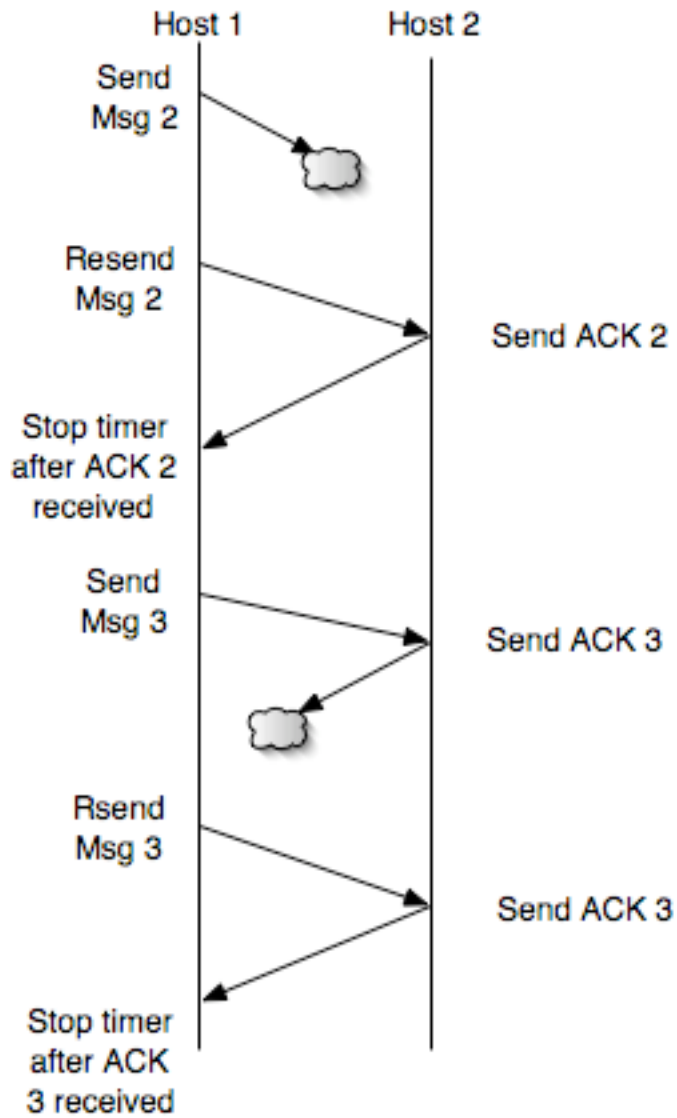
when expected data does not arrive

2. Retransmission

- The sender starts a timer when a message is transmitted
- If the timer expires before an acknowledgment arrives, the sender retransmits the message



- Bigger example: (shows dropped message and dropped acknowledgment)



How long should TCP wait before retransmitting?

- If the timer is too short, you might have more than one msg/ACK in the ether at the same time

- Unneeded retransmissions can happen (excessive retransmission)
- Acknowledgments arrive after timer expires

- Timer can be too long

- Slow throughput in the presence of packet lost
- TCP will wait for too long to retransmit
- Throughput would suffer

- The timer should be long enough to allow the acknowledgment to arrive.

$$T = (1+\alpha)RTT$$

- The timer should be a little bit longer than the round trip time (RTT) to give enough time to acknowledgments to arrive
- TCP adaptive retransmission keeps an estimate of the roundtrip time
- TCP uses this estimate to compute the retransmission timer as follows

$$T = RTT + \text{stddev}(RTT)$$

("retransmission timer" = "Estimated average of round trip time" + "standard deviation of the RTT")

The standard deviation of the RTT takes into account the variability of the round-trip-time with respect to the average.

- round-trip-time(T) is affect by:
 - distance to destination
 - number of routers and processing time
 - transmission medium
 - current traffic conditions ***
- RTT average may change over time. TCP re-estimates RTT after every successful transmission.

Only packets that do not receive retransmission contribute to RTT-Estimate

- The RTT average and RTT-standard-deviation are estimated by TCP using integer arithmetic to make it fast
- TCP uses "moving average" to compute the RTT average and standard deviation .

Moving average gives more weight to the newer samples than the old samples

$$RTT_{avg}(t+1) = aRTT_{sample} + bRTT_{avg}(t)$$

$$a + b = 1$$

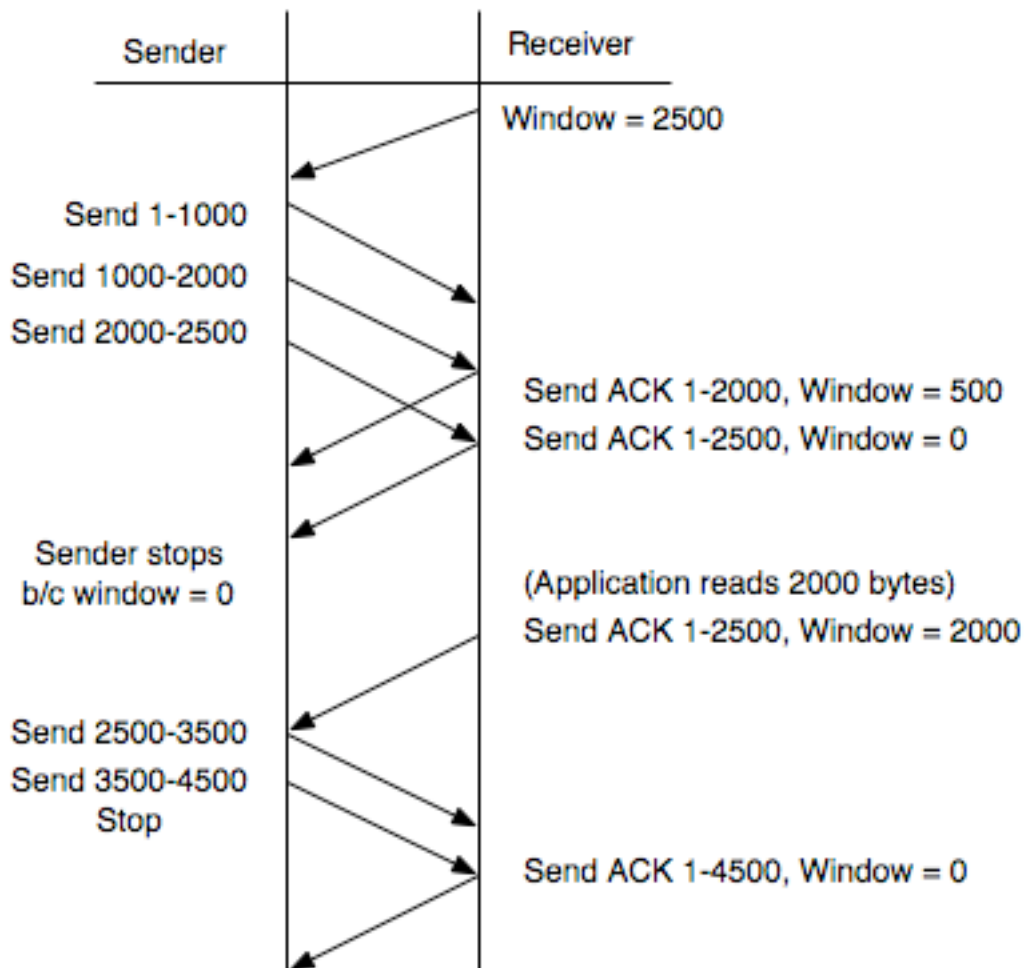
$$\text{eg: } a = .2, b = .8$$

- Adaptive retransmission is a key for TCP success since it allows TCP to run in fast networks as well as slow networks

TCP Flow Control

- The sender should slow down transmission if the receiver application is not fast enough to consuming the data sent
- Example: a fast computer sends 20MB of data but the receiver has only 8KB of buffer space. The sender should stop sending when the buffer is full
- To do this, the receiver advertises to the sender the available buffer space in the receiver
- The sender can only send the advertised number of bytes of buffer space in the receiver and then stop
- The available buffer space in the receiver is called a "window"

- The receiver advertises the window size in each acknowledgment
- The sender cannot send more data than the advertised window size
- Each acknowledgment carries the following "I have received up to X bytes correctly and I can take Y more"
- In TCP each packet is not acknowledged individually. Instead, the acknowledgments contain the total number of bytes that have been received without a gap (continuous bytes)
- Also, TCP does not wait for an acknowledgment to send more data, but it sends multiple packets before an ACK is received



- TCP acknowledges all the data in sequence that has been received so far
 - eg: ACK byte 1-4500
 - This allows TCP not to send an ACK for every packet received. It reduces the # of acknowledgments needed.

