Transmission Control Protocol

Agenda

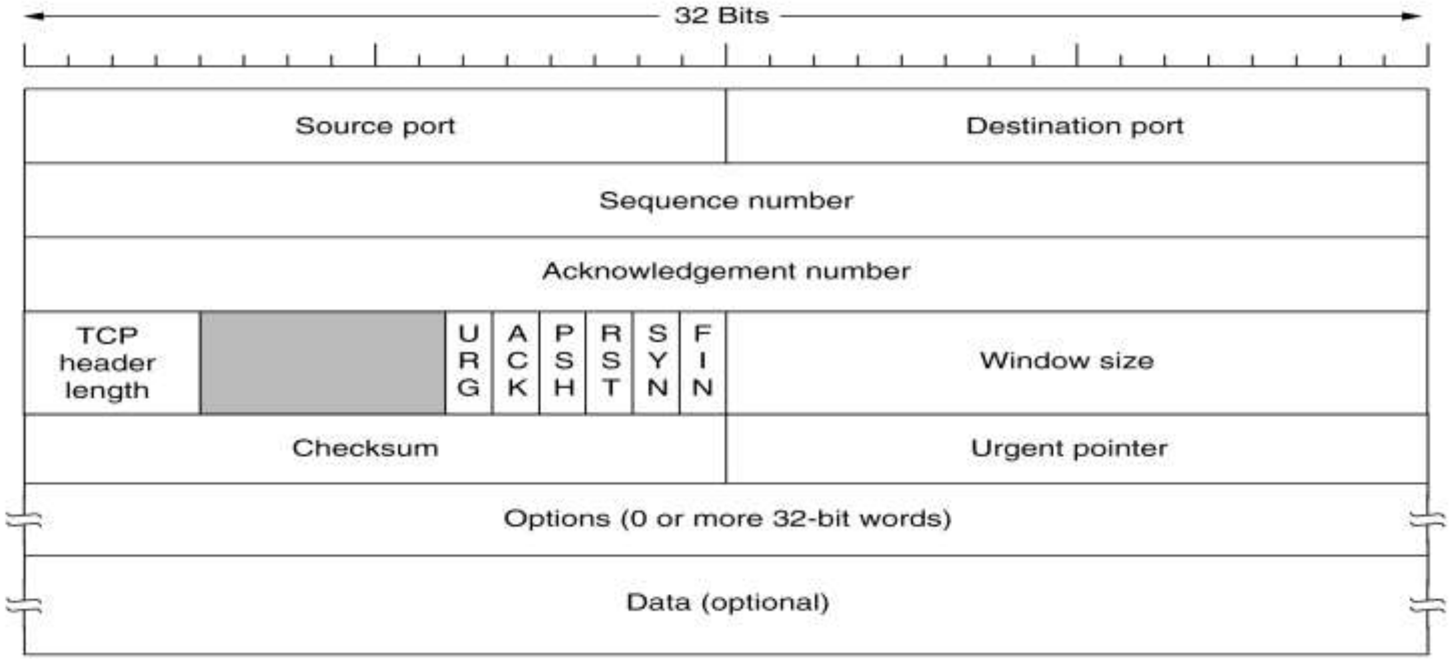
- Introduction
- Features
- Connection Establishment and Termination
- Flow Control
- Error Control
- Congestion Control

TCP Introduction

- TCP:
  - Process-to-Process communication
  - Full Duplex communication
  - Connection-oriented service
  - Stream-oriented protocol (stream of bytes)
  - Segmentation (datagrams in IP layer)
  - Utilizes buffers at both ends (flow & error control)
  - Reliable (ACKs)

- TCP seeks to deliver a byte stream from end-to-end, in order, reliably.

TCP Header


32 Bits
Source port
Destination port
Sequence number
Acknowledgement number
TCP header length
URG
ACK
PSH
RST
SYN
FIN
Window size
Checksum
Urgent pointer
Options (0 or more 32-bit words)
Data (optional)

TCP Features

- Numbering system
  - Byte Number
    - TCP numbers all the data bytes that are transmitted in a connection
    - Numbering does not necessarily start from 0
    - TCP generates a random number between 0 and 2^{32}-1 for numbering first byte
    - Eg. Random no. = 3423 (first byte)
    - Total bytes = 5000
    - Numbering range = 3423-8422

TCP Features

- Sequence number
  - Bytes are grouped into "segments"
  - Sequence number for each segment is the number of the first byte carried in that segment
  - 3423 (3423-4422)
  - 4423 (4423-5422)
  - 5423 (5423-6422)
  - 6423 (6423-7422)
  - 7423 (7423-8422)

TCP Features

- Acknowledgement number
  - Defines the number of the next byte that the party expects to receive
  - It is cumulative
  - ACK = 5487
  - It means it has received all bytes from beginning up to 5486 (beginning may not be 0)

TCP Features

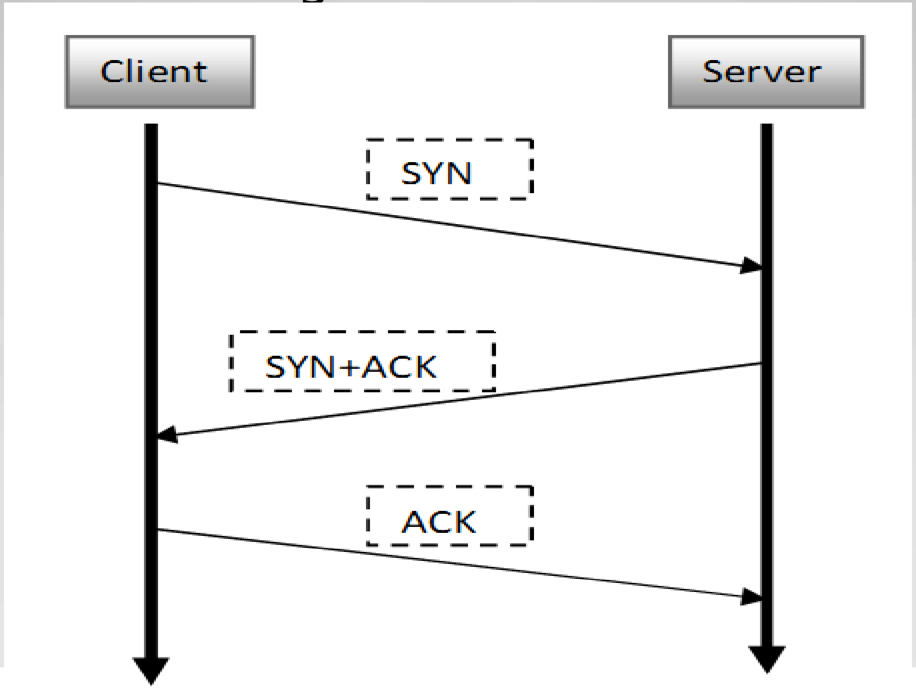
- Flow control
  - Receiver of the data controls the amount of data that are to be sent by the sender
  - Numbering system allows TCP to use byte-oriented flow control
- Error control
  - Considers a segment as a unit of data for error detection
- Congestion control

Connection Establishment

- Establishes a virtual path between the source and destination
- How TCP is connection-oriented while using IP (connection-less)?
  - Connection is virtual
  - TCP uses the services of IP to deliver individual segments, but it controls the connection itself
  - IP is unaware of retransmission, out-of-order segments

Connection Establishment

- Three way Handshaking

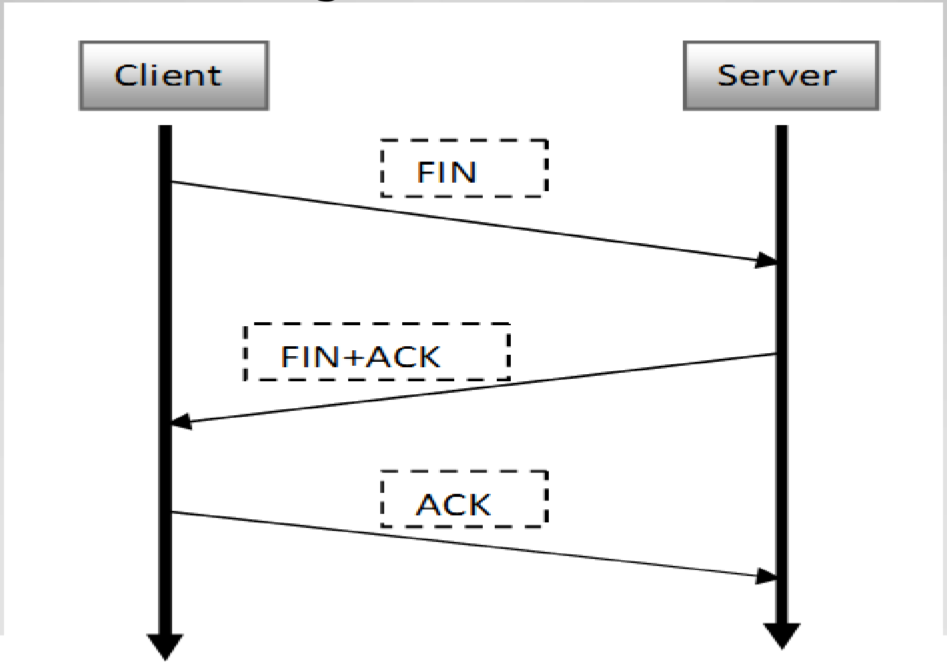


Connection Establishment

- SYN:
  - It is for synchronization of sequence numbers
  - It consumes 1 sequence number
  - Carries no real data
- SYN+ACK:
  - SYN segment for communication in other direction and ACK for the received SYN
  - It consumes 1 sequence number
- ACK
  - Just an ACK segment
  - Does not consume any sequence number

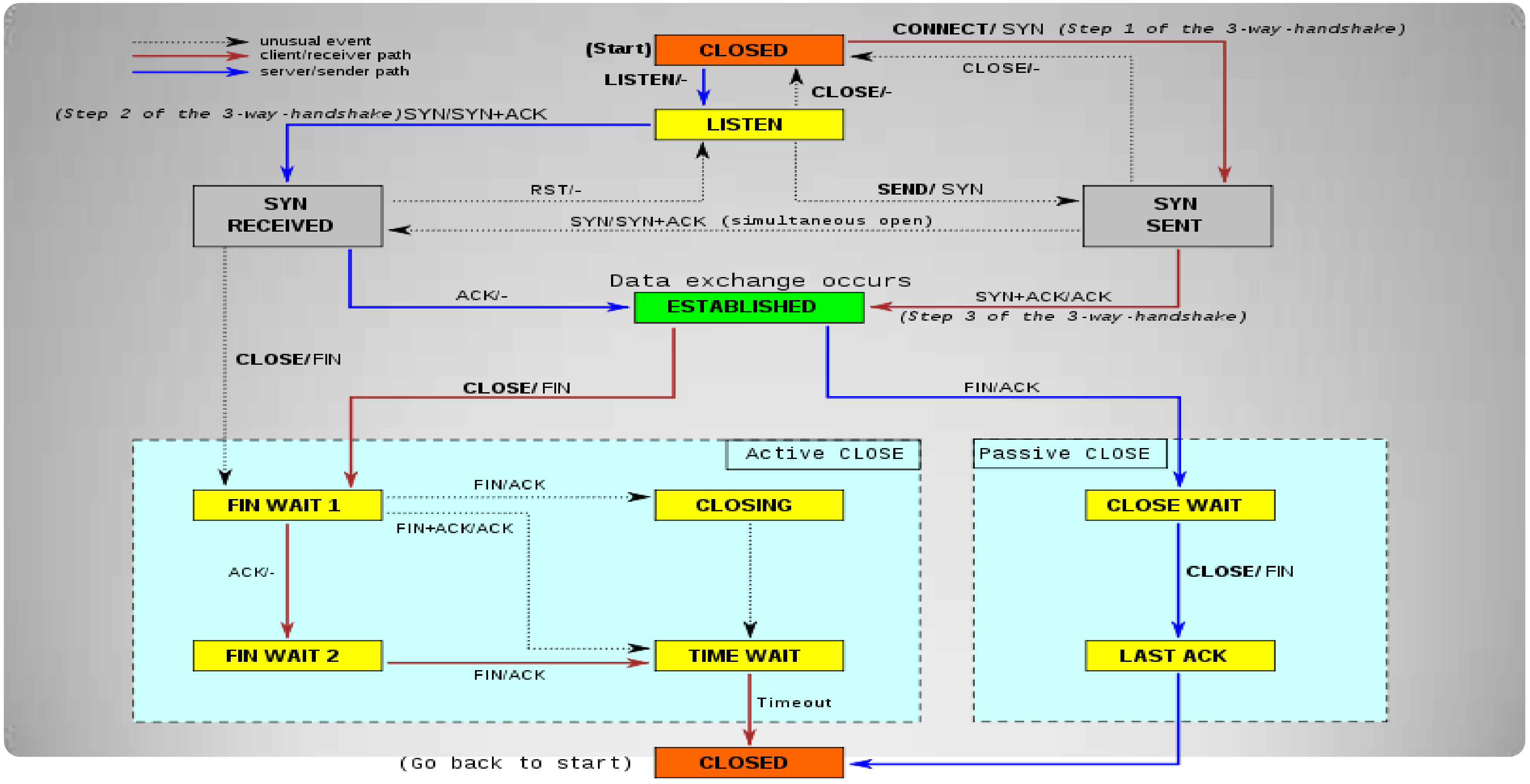
Connection Termination

- Three way Handshaking



Connection Termination

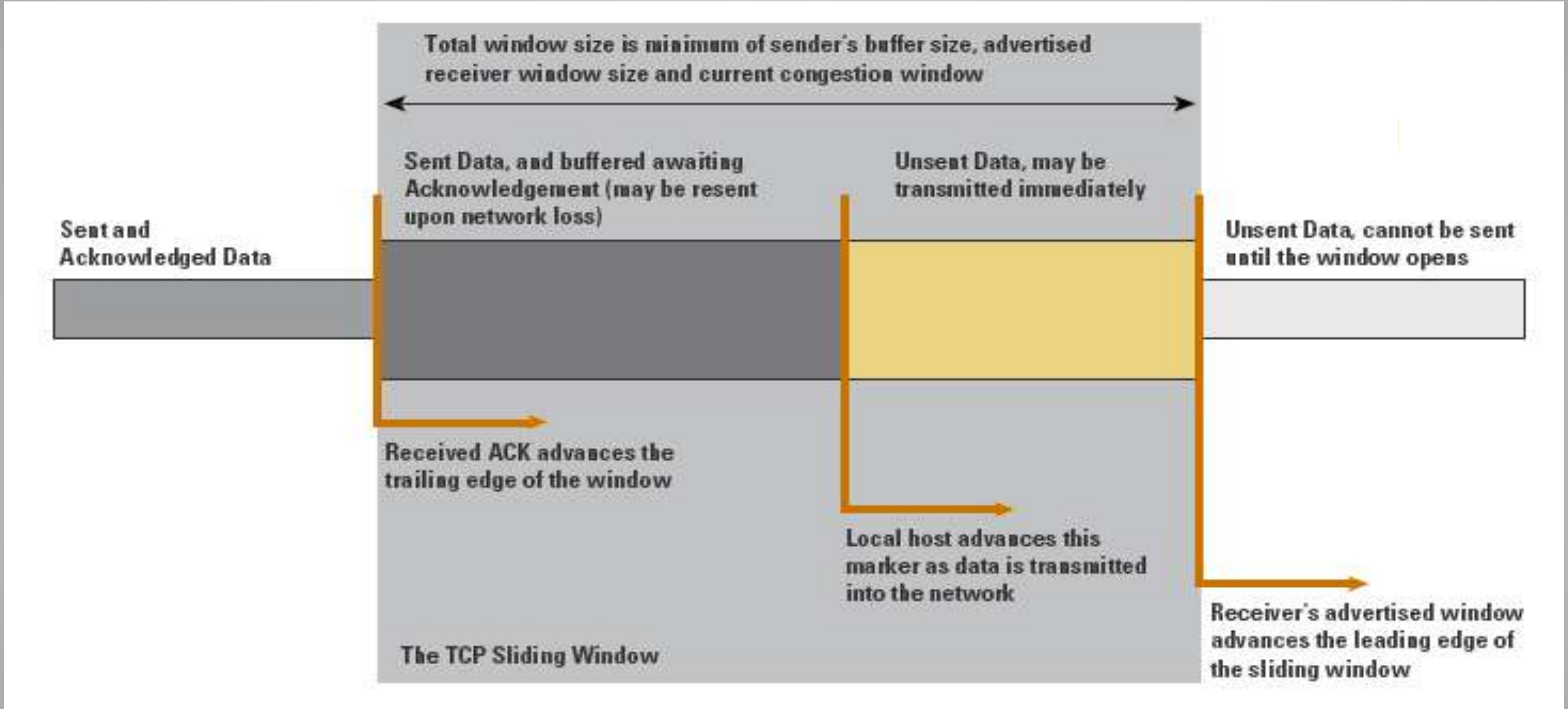
- FIN:
  - It consumes 1 sequence number
  - May or may not carry real data
- FIN+ACK:
  - FIN segment to announce closing of connection in other direction and ACK for the received FIN
  - It consumes 1 sequence number
- ACK
  - Just an ACK segment
  - Does not consume any sequence number


unusual event
client/receiver path
server/sender path
(Start)
CLOSED
CONNECT/ SYN (Step 1 of the 3-way-handshake)
CLOSE/-
LISTEN/-
CLOSE/-
(Step 2 of the 3-way-handshake)SYN/SYN+ACK
LISTEN
SYN RECEIVED
RST/-
SEND/ SYN
SYN SENT
SYN/SYN+ACK (simultaneous open)
Data exchange occurs
ACK/-
ESTABLISHED
SYN+ACK/ACK
(Step 3 of the 3-way-handshake)
CLOSE/ FIN
CLOSE/ FIN
FIN/ACK
Active CLOSE
Passive CLOSE
FIN WAIT 1
FIN/ACK
CLOSING
FIN+ACK/ACK
CLOSE WAIT
ACK/-
CLOSE/ FIN
FIN WAIT 2
FIN/ACK
TIME WAIT
LAST ACK
Timeout
(Go back to start)
CLOSED

Flow Control

- TCP uses sliding window to handle flow control
- The size of the window is determined by the lesser of two values: *rwnd* or *cwnd*
- *rwnd:* it is the number of bytes the receiver can accept before its buffer overflows
- *cwnd:* it is the value determined by the network to avoid congestion
- The receiver controls most of the aspects

Flow Control


Total window size is minimum of sender's buffer size, advertised receiver window size and current congestion window
Sent and Acknowledged Data
Sent Data, and buffered awaiting Acknowledgement (may be resent upon network loss)
Unsent Data, may be transmitted immediately
Unsent Data, cannot be sent until the window opens
Received ACK advances the trailing edge of the window
Local host advances this marker as data is transmitted into the network
Receiver's advertised window advances the leading edge of the sliding window
The TCP Sliding Window

Error Control

CAR

- Includes mechanisms for detecting corrupted segments, lost segments, out-of-order segments and duplicated segments
- Achieved through the use of three simple tools:
  - Checksum
  - Acknowledgement
  - Retransmission

CAR

Checksum

- Each segment includes a checksum field, used to check for corrupted segment
- TCP uses a 16-bit checksum
- Corrupted segment is discarded by the destination and is considered lost

Acknowledgement

- Confirm the receipt of data segments
- Control segments that carry no data but consume a sequence number are also acknowledged
- ACK segments are never acknowledged

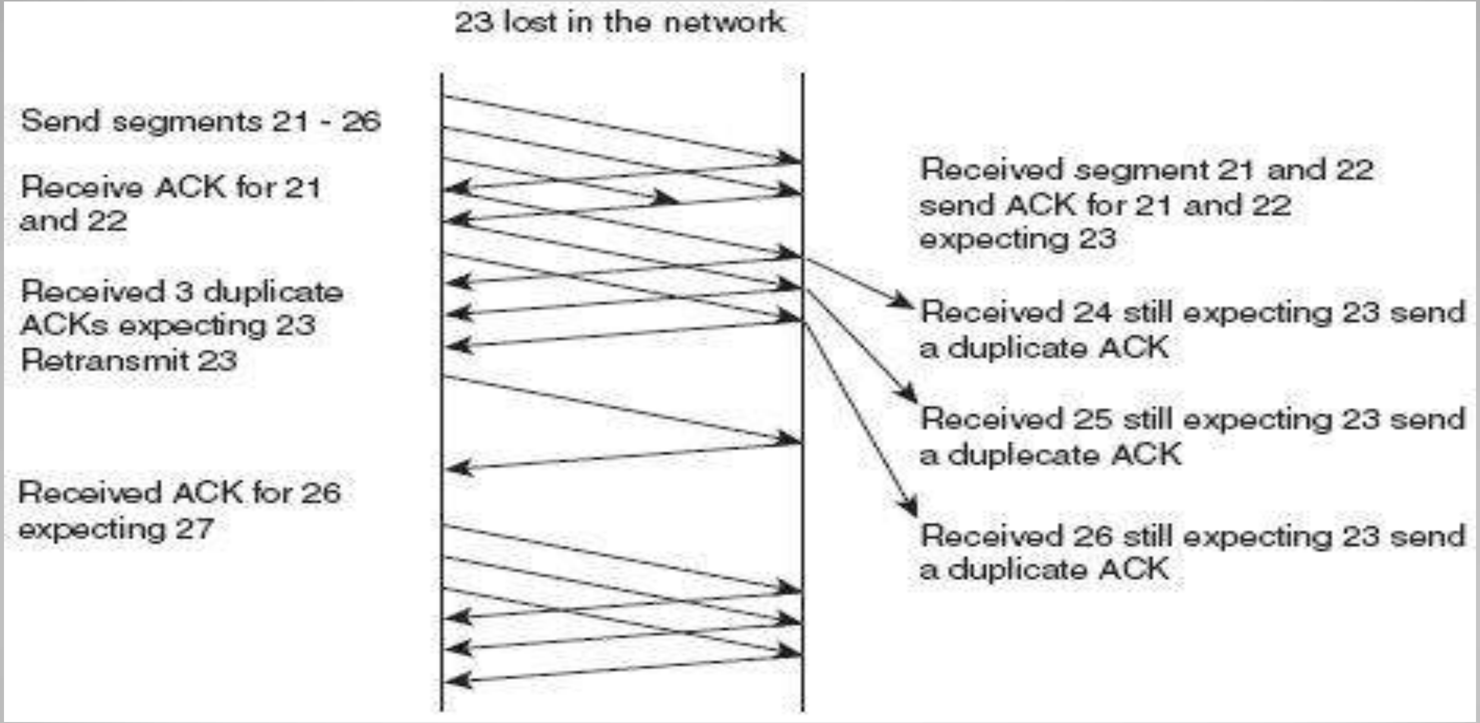
Retransmission

- A segment is retransmitted on two occasions:
  - When a retransmission timer expires
  - When the sender receives three duplicate ACKs
- There is no retransmission for ACK segments
- Retransmission after RTO:
  - TCP maintains one RTO timer for all outstanding (sent, but not acknowledged) segments
  - When the timer matures, the earliest outstanding segment is retransmitted
  - Value of RTO is dynamic and is updated based on RTT

Fast Retransmission

- Let the value of RTO be very large
- One segment is lost and receiver receives so many out-of-order segments that they cannot be saved (buffer size)
- When the sender receives 4 ACKs with same value (1 original and 3 duplicates), even though the timer has not matured the fast retransmission requires that the segment be resent immediately

Fast Retransmission


23 lost in the network
Send segments 21 - 26
Receive ACK for 21 and 22
Received 3 duplicate ACKs expecting 23 Retransmit 23
Received ACK for 26 expecting 27
Received segment 21 and 22 send ACK for 21 and 22 expecting 23
Received 24 still expecting 23 send a duplicate ACK
Received 25 still expecting 23 send a duplecate ACK
Received 26 still expecting 23 send a duplicate ACK

Fast Retransmission

- When the sender receives retransmitted ACK, it knows that the four segments are safe and sound because ACK is cumulative

Adaptive Retransmission

- TCP attempts to determine the approximate round-trip time between the devices, and adjusts it over time to compensate for increases or decreases in the average delay.
- TCP aims for an *average* RTT value for the connection.
- This average should respond to consistent movement up or down in the RTT without overreacting to a few very slow or fast acknowledgments.
- TCP re-estimates RTT after every successful transmission (not retransmission).

Adaptive Retransmission

- The RTT calculation uses a *smoothing* formula:
  - New RTT = (a * Old RTT) + ((1-a) * Newest RTT Measurement)
- Where “a” (alpha) is a *smoothing factor* between 0 and 1.
- Higher values of “a" (closer to 1) provide better smoothing and avoiding sudden changes as a result of one very fast or very slow RTT measurement

Adaptive Retransmission

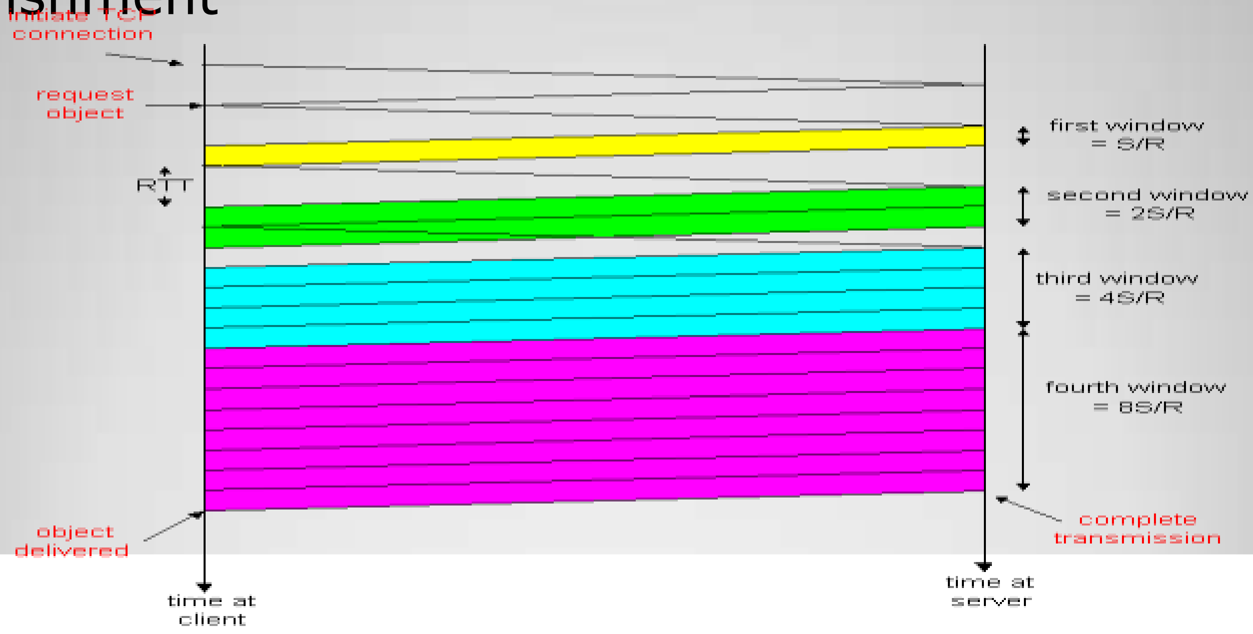
- Conversely, this also slows down how quickly TCP reacts to more sustained changes in round-trip time.
- Lower values of alpha (closer to 0) make the RTT change more quickly in reaction to changes in measured RTT, but can cause “over-reaction” when RTTs fluctuate wildly.
- Adaptive retransmission is a key for TCP success since it allows TCP to run in fast networks as well as slow networks.

Congestion Control

- TCP’s general policy for handling congestion is based on three phases:
  - Slow Start: Exponential Increase
  - Congestion Avoidance: Additive Increase
  - Congestion Detection: Multiplicative Decrease

Slow Start

- Size of congestion window (cwnd) starts with 1 max. segment size (MSS), determined during conn. establishment



Slow Start

- Slow start cannot continue indefinitely
- Sender keeps a track of a variable named *ssthresh*, when the size of window, in bytes, reaches this threshold, slow start stops
- In most cases the value of *ssthresh* is 65,535 bytes

Congestion Avoidance

- When the slow start phase stops, the additive phase begins
- Each time the whole window of segments is acknowledged, the size of the congestion window is increased by 1

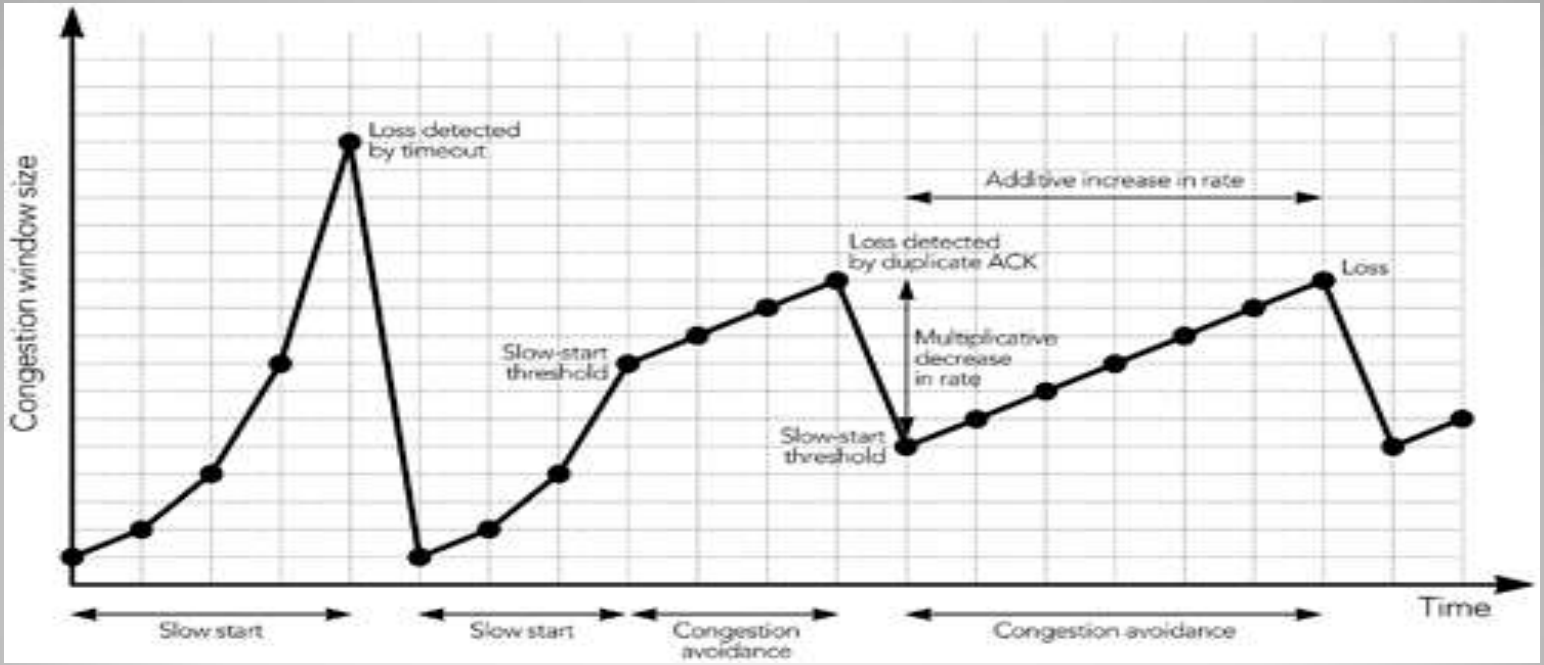
Congestion Detection

- If congestion occurs, the congestion window size must be decreased
- Sender can guess congestion by need to retransmit a segment
- Retransmission can occur in one of two cases:
  - Time-out
  - 3 ACKs are received

Congestion Detection

- If detection is by time-out:
  - It sets the value of the threshold to one half of current window size
  - It sets cwnd to the size of one segment
  - It starts the slow-start phase again
- If detection is by 3 ACKs:
  - It sets the value of the threshold to one half of current window size
  - It sets cwnd to the value of the threshold
  - It starts the congestion avoidance phase

Congestion Example



Questions?

Thank You