Transport Layer Protocols TCP/IP Protocol Suit

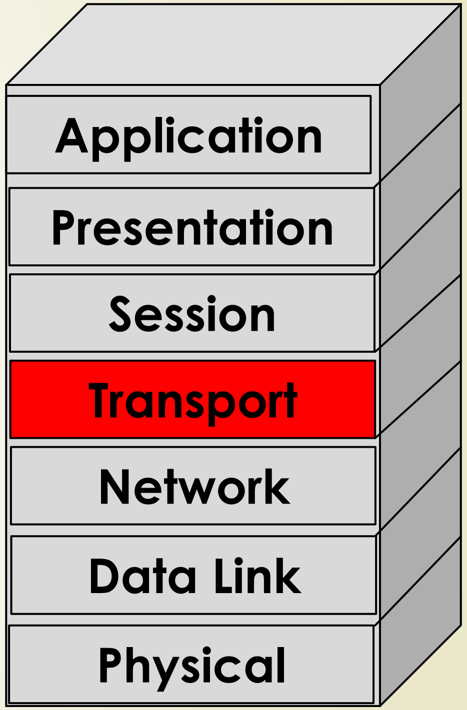
- Key function of the Transport Layer Protocols
- Introduction to Transport Protocols Similar to TCP
- TCP/IP Protocol Suit (part-1)
    - The Service TCP Provides to Applications
    - TCP Header Structure
    - TCP Congestion Control Algorithms
- The Internet Protocol (IP) (part-2)
    - IP Addresses
    - IP Header Structure

Transport Layer Protocols

The following are some additional features of transport layer protocols:

- Across the communication networks, it deals with the flow-control of data between different peers.
- It provides the facility of error detecting, error correcting and assurances for free of error transfer of data over the networks.
- It guarantees a reliable and efficient data transmission through sending acknowledgements (ACKs) for previously data packets collected.
- It manages and handles the network data-flow, if network congestion is encountered; it utilizes effective congestion control mechanisms to maintain its data-rate.
- It is responsible for the retransmission of dropped or lost packets during communication.

OSI Reference Model



Transport Protocols Similar to TCP

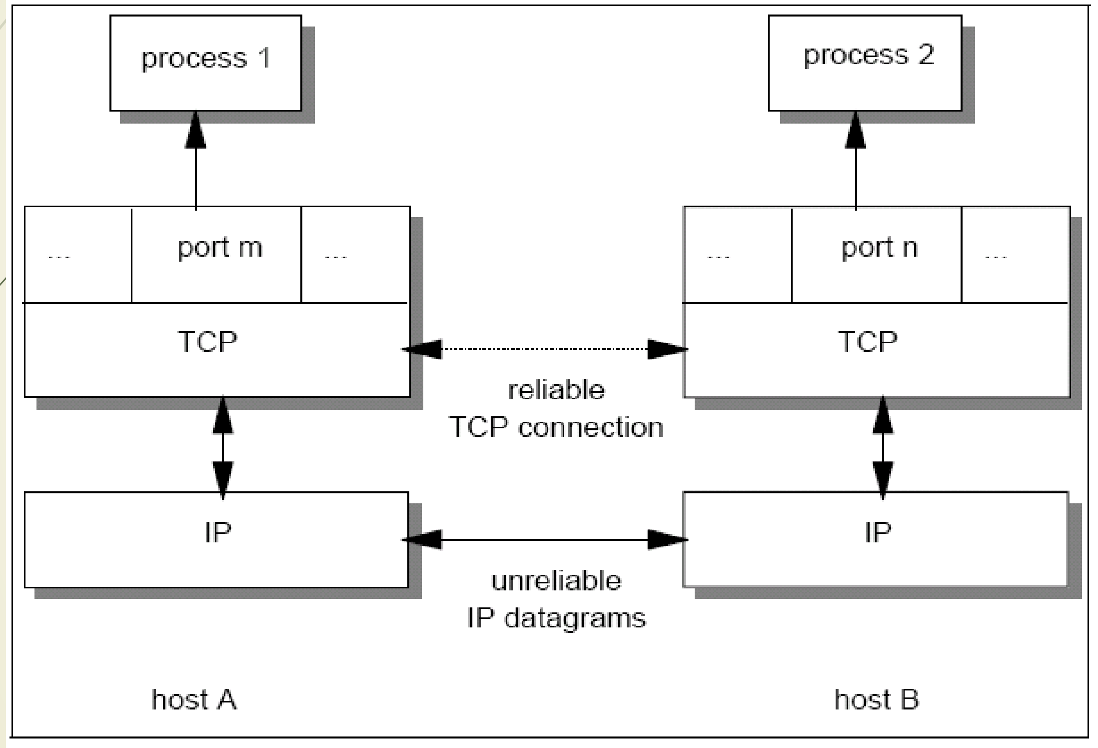
Protocols Name	Mechanism/Approach	Protocol's Purpose/Functionality
TCP New Reno	Standard Additive Increase, Multiplicative Decrease (AIMD) Algorithm and Loss-Based Approach	"Standard Transmission Control Protocol (TCP)"
Stream Control Transmission Protocol (SCTP)	Standard AIMD Algorithm and Loss-Based Approach	For message-oriented applications, the core original features of SCTP are multistreaming and multihoming
TCP Vegas	Multiplicative Increase, Multiplicative Decrease (MIMD) Algorithm and Delay-Based Approach	Higher throughput and reduced loss rate
FAST TCP	MIMD Algorithm and Delay-Based Approach	Modified TCP Vegas. Higher throughput with high capacity and large delays
Scalable TCP	Modified AIMD Algorithm and Loss-Based Approach	Great aggressive in Congestion Control. Higher throughput with large delays and high-capacity
BIC-TCP	Modified AIMD Algorithm, binary search technique is combined with additive increase and Loss-Based Approach	A unique window control function is used. Improves the network utilization, and makes it fair with other high speed flows
CUBIC	Modified AIMD Algorithm and Loss-Based Approach	A New TCP-Friendly High-Speed TCP Variant. Higher throughput with high capacity and large delays
High-Speed TCP	Modified High-Speed AIMD Algorithm and Loss-Based Approach	High throughput with high bandwidth and large propagation delays. Adjust TCP response function when cwnd is higher than a threshold value
eXplicit Congestion Control Protocol (XCP)	Similar principle that TCP uses (AIMD) to converge to fairness and Extra Signaling Approach	It uses precise congestion signaling, and fairness control. Higher throughput and faster convergence with high capacity and high delays. Smaller queues.
TCP-XM	AIMD Algorithm and Loss-Based Approach	Extends TCP to allow sender initiated data transfers to n-known hosts, combining native network IP multicast alongside traditional unicast data transmission
Westwood+	Modified AIMD Algorithm and based on end-to-end bandwidth estimation	Modification of TCP Reno, Proposed for high capacity and large delay networks

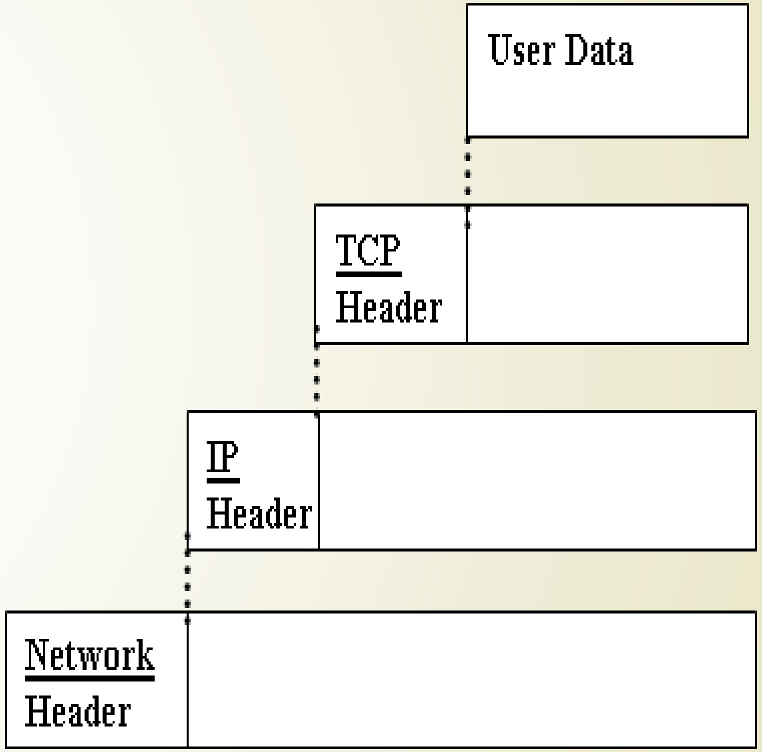
TCP/IP Protocol Suit

The TCP/IP suite includes the following protocols:

- TCP – Transmission Control Protocol
- IPv4 / IPv6 – Internet Protocol

Transmission Control Protocol (TCP)





The Service TCP Provides to Applications

- Connection Orientation
- Point-To-Point Communication
- Complete Reliability
- Full Duplex Communication
- Stream Interface
- Reliable Connection Startup
- Graceful Connection Shutdown

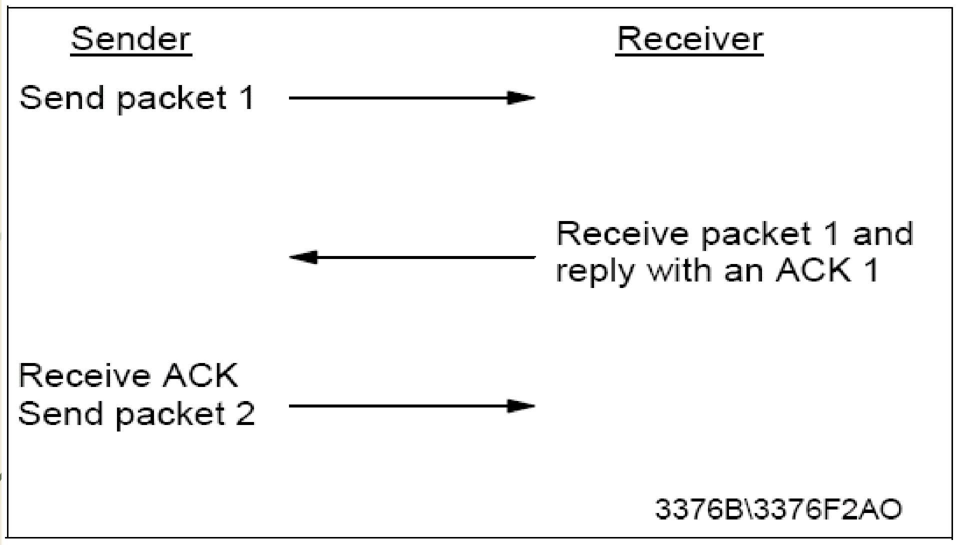
TCP Header Structure

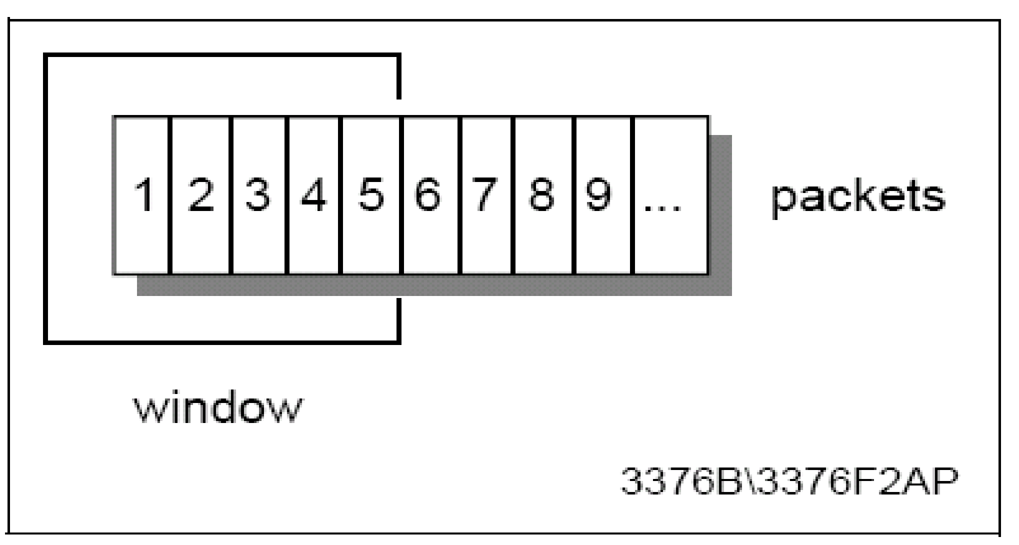
<table>
<tr><td colspan="4">0</td><td colspan="4">15</td><td>31</td></tr>
<tr><td colspan="8">16</td><td>32 bits</td></tr>
<tr><td colspan="8">Source port</td><td>Destination port</td></tr>
<tr><td colspan="9">Sequence number</td></tr>
<tr><td colspan="9">Acknowledgement number</td></tr>
<tr><td>Offset</td><td>Resrvd</td><td>U</td><td>A</td><td>P</td><td>R</td><td>S</td><td>F</td><td>Window</td></tr>
<tr><td colspan="8">Checksum</td><td>Urgent pointer</td></tr>
<tr><td colspan="9">Option + Padding</td></tr>
<tr><td colspan="9">Beginning of Data
.
.</td></tr>
</table>

Definitions

- Segment
- Sender Maximum Segment Size (SMSS)
- Receiver Maximum Segment Size (RMSS)
- Receiver Window (RWND)
- Congestion Window (CWND)
- Loss Window (LW)
- Restart Window (RW)

TCP Window Principle





TCP Window Principle

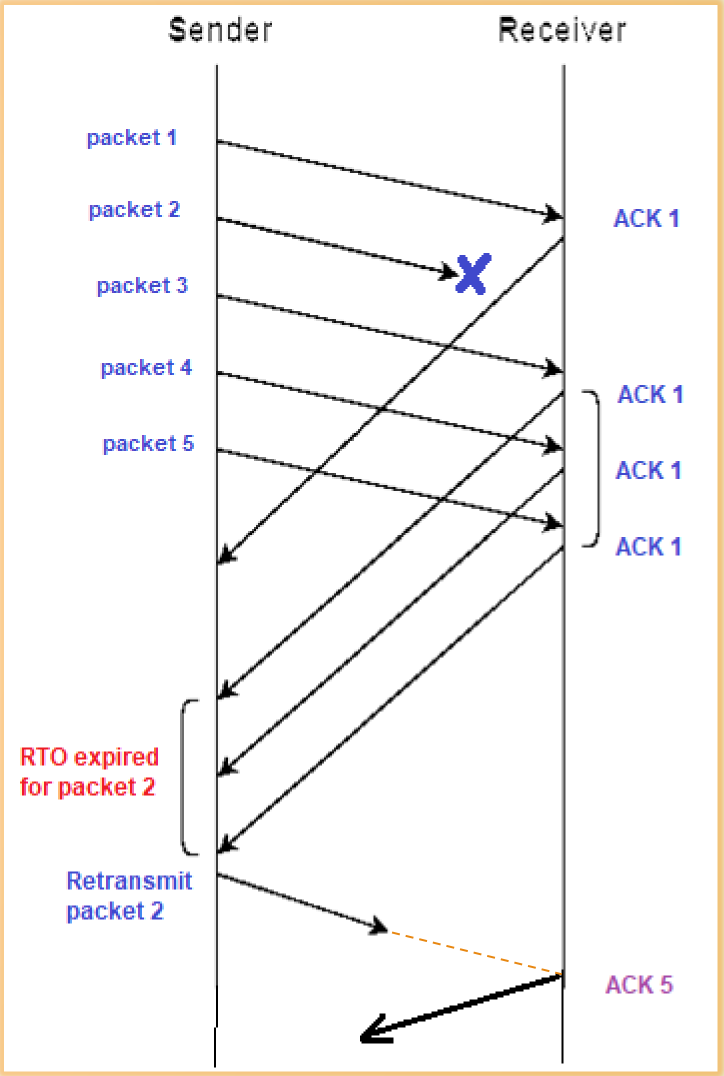
Imagine some special cases:

Packet 2 gets lost

- sender will not receive ACK 2, window will remain in position 1
- receiver did not receive packet 2, it will acknowledge packets 3, 4, and 5 with an ACK 1
- timeout will occur for packet 2 and it will be retransmitted, receiver will generate ACK 5
- successfully received all packets 1 to 5, sender's window will slide four positions upon receiving this ACK 5

Packet 2 did arrive, but the acknowledgment gets lost

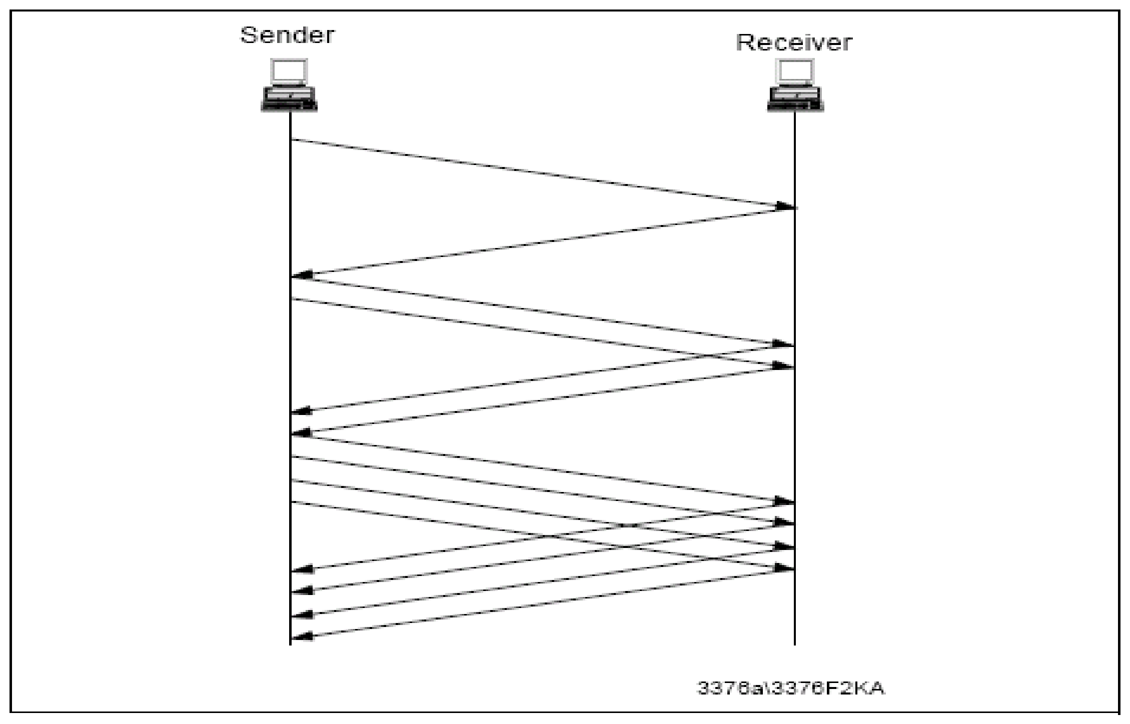
- sender does not receive ACK 2, but will receive ACK 3
- ACK 3 is an acknowledgment for all packets up to 3 (including packet 2)
- sender can now slide its window to packet 4



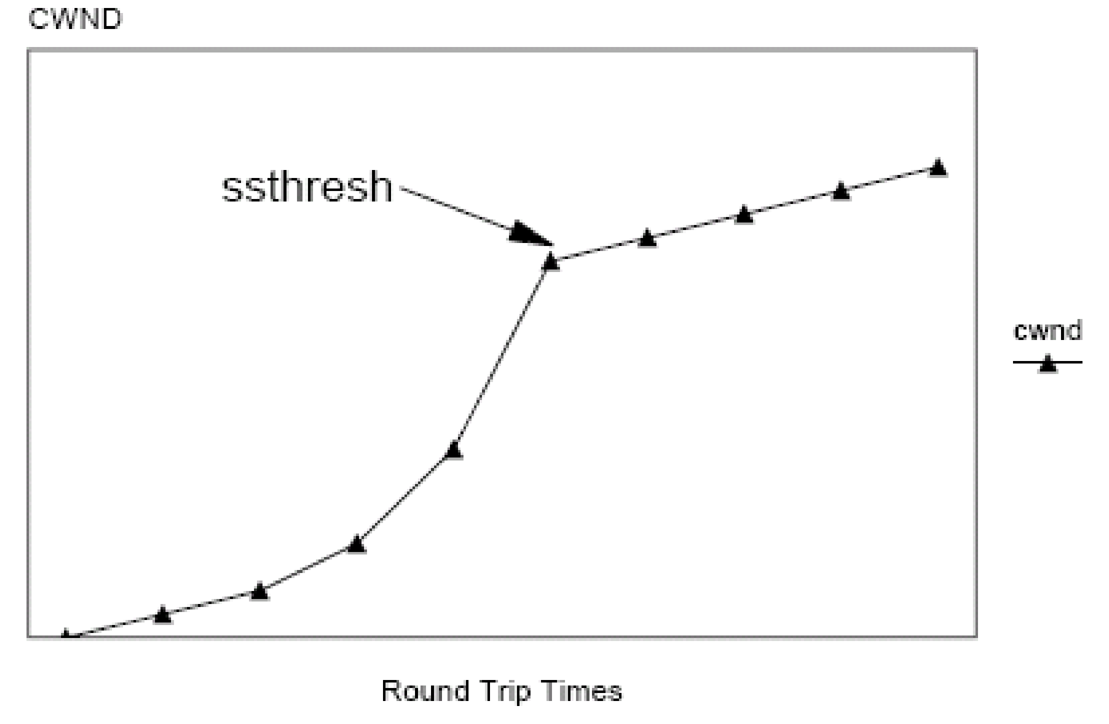
TCP Congestion Control Mechanism

- Slow Start and Congestion Avoidance
- Fast Retransmit/Fast Recovery

Slow Start and Congestion Avoidance



TCP Slow Start

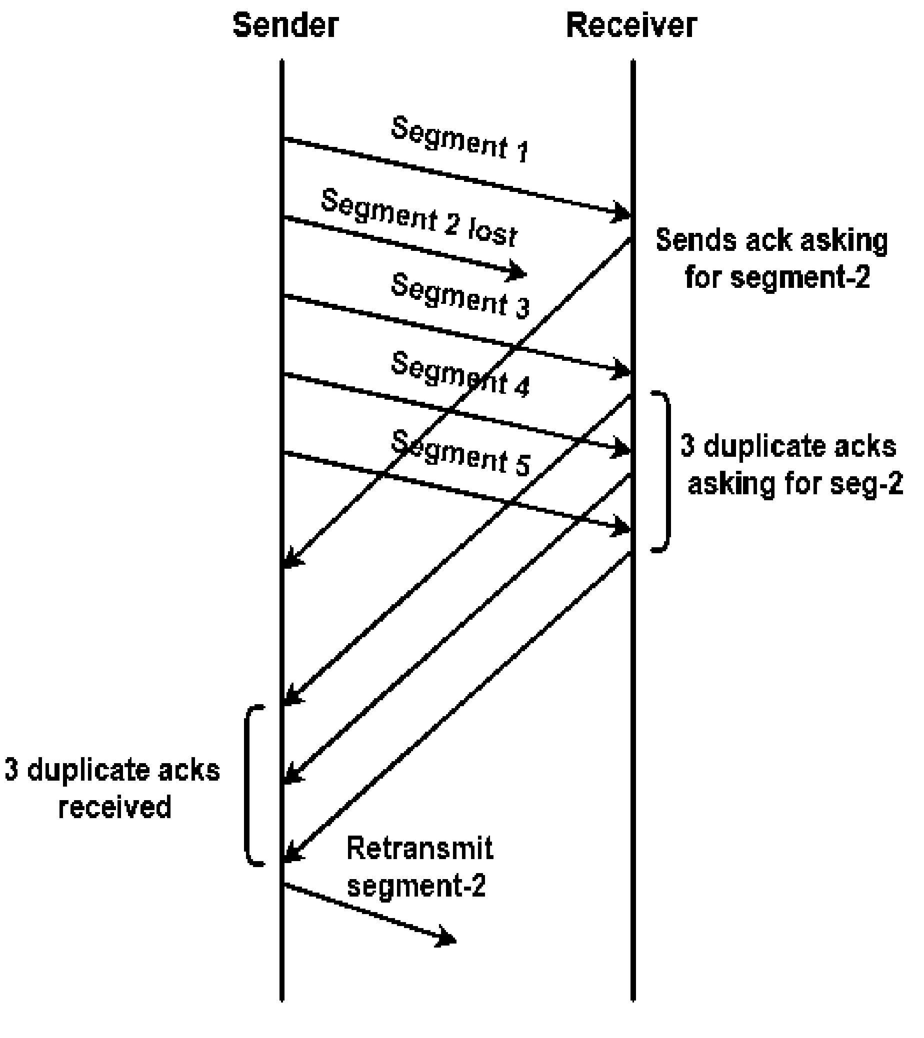


TCP Slow Start & Congestion Avoidance

Fast Retransmit/Fast Recovery

Three duplicate ACK received:

- retransmit the packet
- ssthresh = cwnd/2
- cwnd = ssthresh + 3 packets.



Retransmission after receiving 3 duplicate acks

TCP Congestion Control Algorithms

1. cwnd $\leftarrow$ one segment and ssthresh $\leftarrow$ 65535 bytes.
2. Send segments of the minimum of cwnd and the receiver's advertised window.
3. When congestion occurs, ssthresh $\leftarrow$ cwnd / 2, and if the congestion is indicated by a timeout, cwnd $\leftarrow$ one segment (i.e., slow start).
4. When new data is acknowledged, cwnd $\leftarrow$ cwnd + 1 segment size if cwnd $\leq$ ssthresh, or cwnd $\leftarrow$ cwnd + (1 / cwnd) if cwnd $>$ ssthresh.

Fig. 1. The combined algorithm of slow start and congestion avoidance.

1. When the third duplicate ACK is received, ssthresh $\leftarrow$ cwnd / 2, retransmit the missing segment, and cwnd $\leftarrow$ ssthresh + 3 times the segment size.
2. Each time another duplicate ACK arrives, cwnd $\leftarrow$ cwnd + 1 segment size.
3. When a new ACK arrives, cwnd $\leftarrow$ ssthresh.

Fig. 2. The combined algorithm of fast retransmit and fast recovery.

Thank You

Q & A