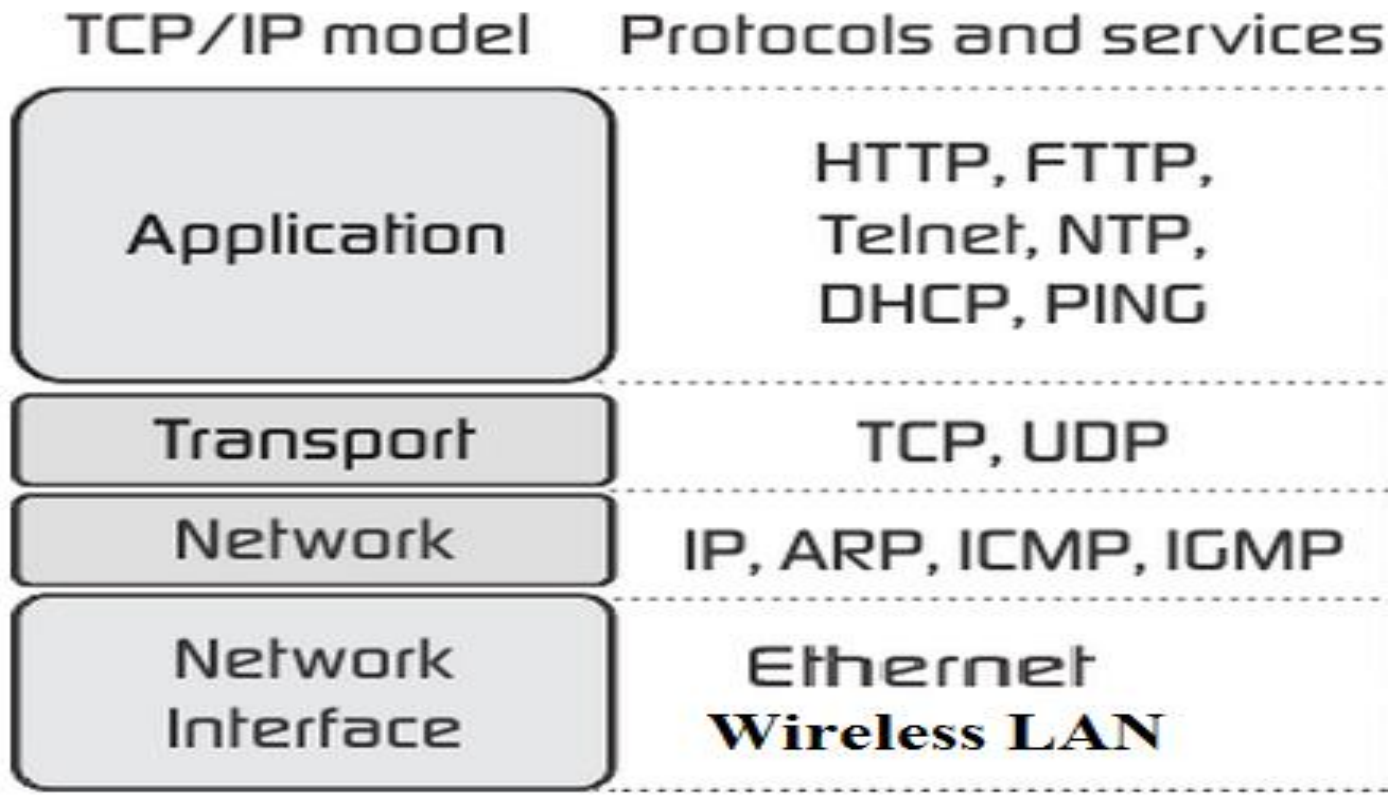
TRADITIONAL TCP

Dr. A. Beulah

AP/CSE

Architecture of TCP/IP

- TCP
  - Telnet, SMTP, FTP
  - HTTP
- UDP
  - DNS
  - SNMP



Terminologies of TCP/IP

- TCP
- IP
- HTTP
- SMTP
- MIME
- FTP
- SNMP

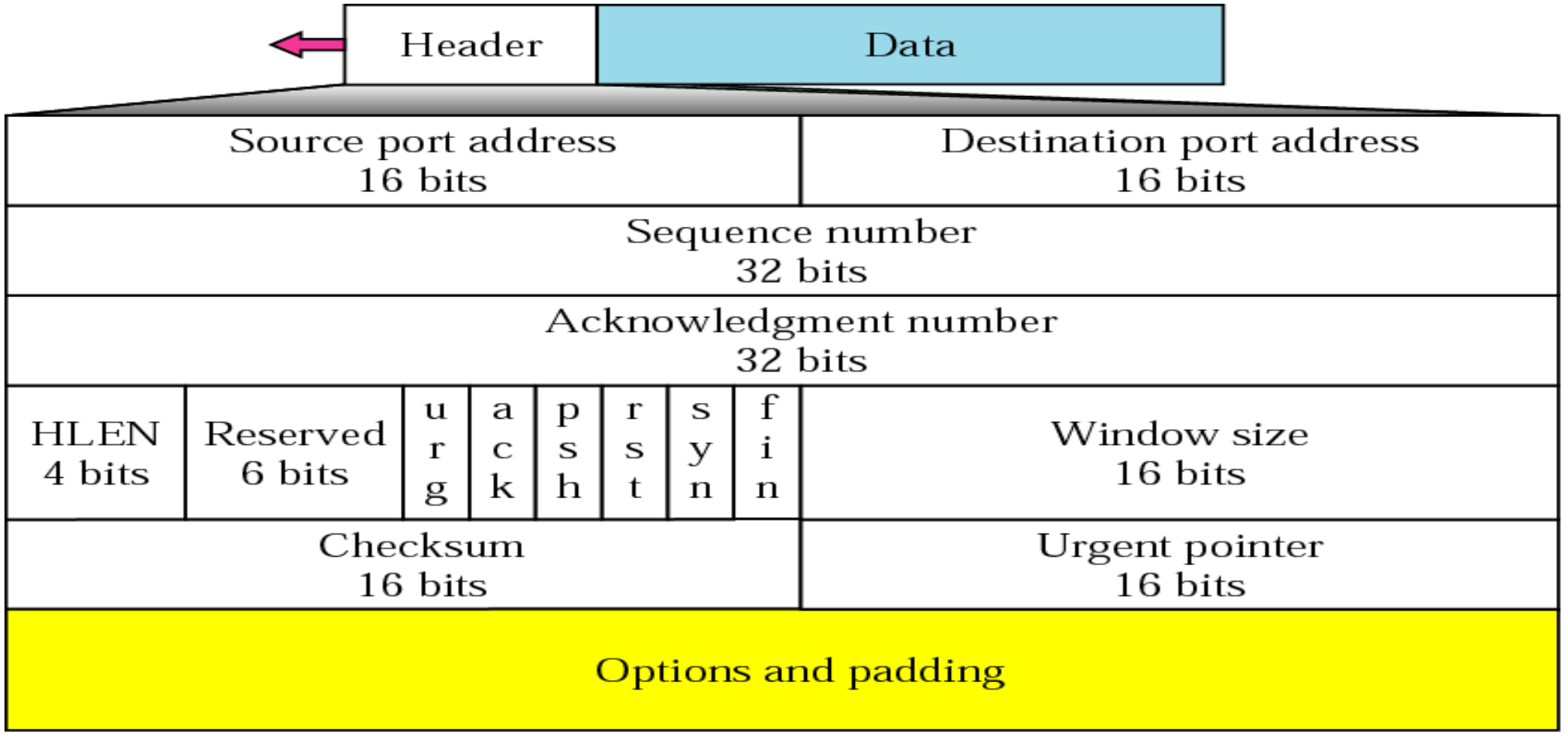
- ARP
- RARP
- BOOTP
- Routers
- DNS
- IP Address
- ICMP
- IGMP

Overview of TCP Operations

- TCP Segment
- Port address
- Data Encapsulation

TCP Segment

- 20 or 60 byte header



Numbering System

- Byte Number
- Sequence Number
- Acknowledgment Number

Byte Number

- Each byte should be numbered
- Random number – 1057
- Data contains 6000 bytes
- 1057 – 7056

Sequence Number

- Sequence number for each segment is the number of the first byte carried in that segment
- Imagine a TCP connection is transferring a file of 6000 bytes. The first byte is numbered 10010. What are the sequence numbers for each segment if data are sent in five segments with the first four segments carrying 1000 bytes and the last segment carrying 2000 bytes?

Sequence Number

- The following shows the sequence number for each segment:

Segment 1 ==> sequence number: 10,010
(range: 10,010 to 11,009)
Segment 2 ==> sequence number: 11,010
(range: 11,010 to 12,009)
Segment 3 ==> sequence number: 12,010
(range: 12,010 to 13,009)
Segment 4 ==> sequence number: 13,010
(range: 13,010 to 14,009)
Segment 5 ==> sequence number: 14,010
(range: 14,010 to 16,009)

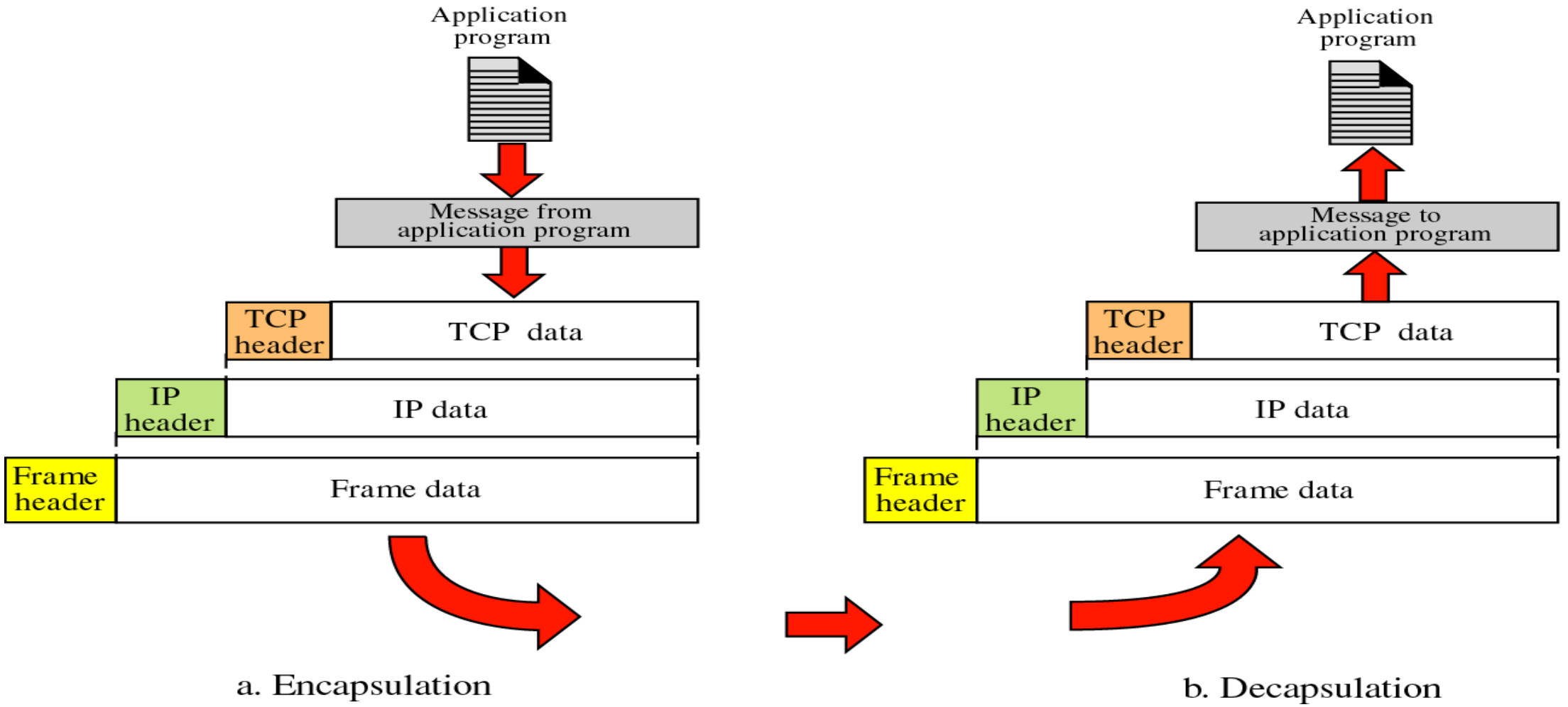
Acknowledgment Number

- The value of the acknowledgment field in a segment defines the number of the next byte a party expects to receive.
- The acknowledgment number is cumulative.

Port Nos.

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

Data Encapsulation



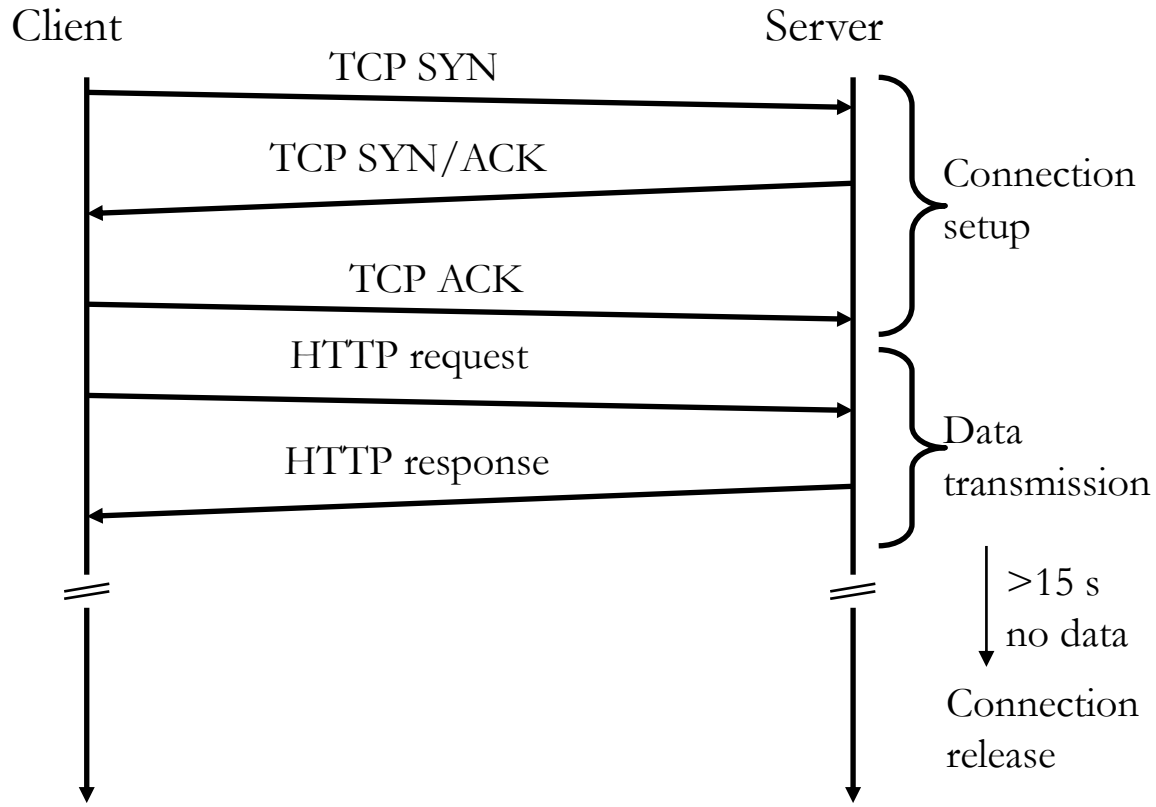
Application Layer Protocols of TCP

- SMTP
- FTP
- Telnet

ADAPTATION OF TCP WINDOW

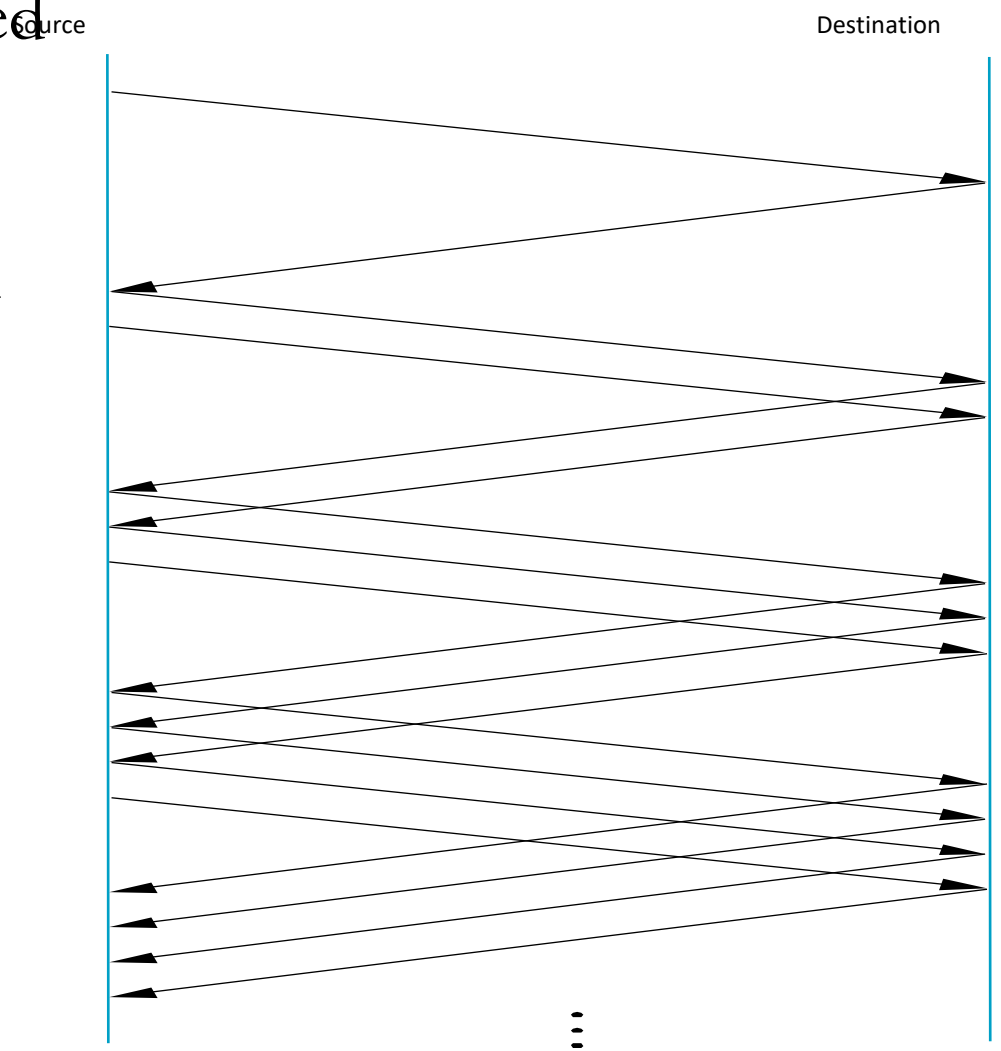
Basic Concepts

- E.g. HTTP (used by web services) typically uses TCP
  - Reliable transport between client and server required
- TCP
  - Stream oriented, not transaction oriented
  - Packet Loss
    - ➔ congestion
    - ➔ slow down transmission
- Well known – TCP guesses quite often wrong in wireless and mobile networks
  - Packet loss due to transmission errors
  - Packet loss due to change of network
- Result
  - Severe performance degradation



Additive Increase

- Additive Increase is a reaction to perceived available capacity.
- Linear Increase → For each "cwnd's worth" of packets sent, increase cwnd by 1 packet.
- In practice, cwnd is incremented exponentially for each arriving ACK.



Silly Window Syndrome

- If a receiver with this problem is unable to process all incoming data, it requests that its sender reduce the amount of data they send at a time.
- MSS/2
- If the receiver continues to be unable to process all incoming data, the window becomes smaller and smaller, sometimes to the point that the data transmitted is smaller than the packet header, making data transmission extremely inefficient.
- The name of this problem is due to the window size shrinking to a "silly" value.
- Nagel's Algorithm

Silly Window Syndrome

Nagel's Algorithm

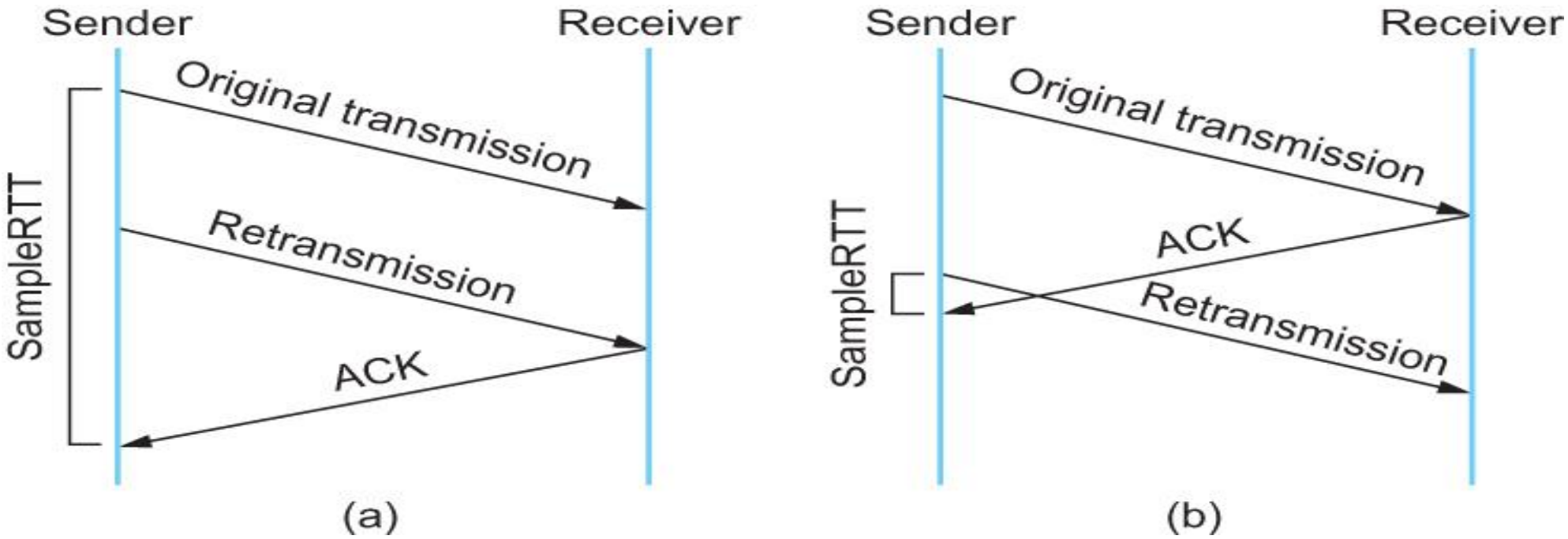
```
if there is new data to send then
    if the window size ≥ MSS and available data is ≥ MSS then
        send complete MSS segment now
    else
        if there is unconfirmed data still in the pipe then
            enqueue data in the buffer until an acknowledge is received
        else
            send data immediately
        end if
    end if
end if
```

Adaptive Retransmission

- TCP achieves reliability by retransmitting segments after a Timeout

- Choosing the value of the Timeout
  - Set time out as a function as RTT
  - If too small, retransmit unnecessarily
  - If too large, poor throughput
  - Make this adaptive, to respond to changing congestion delays in Internet

Adaptive Retransmission



Associating the ACK with (a) original transmission versus (b) retransmission

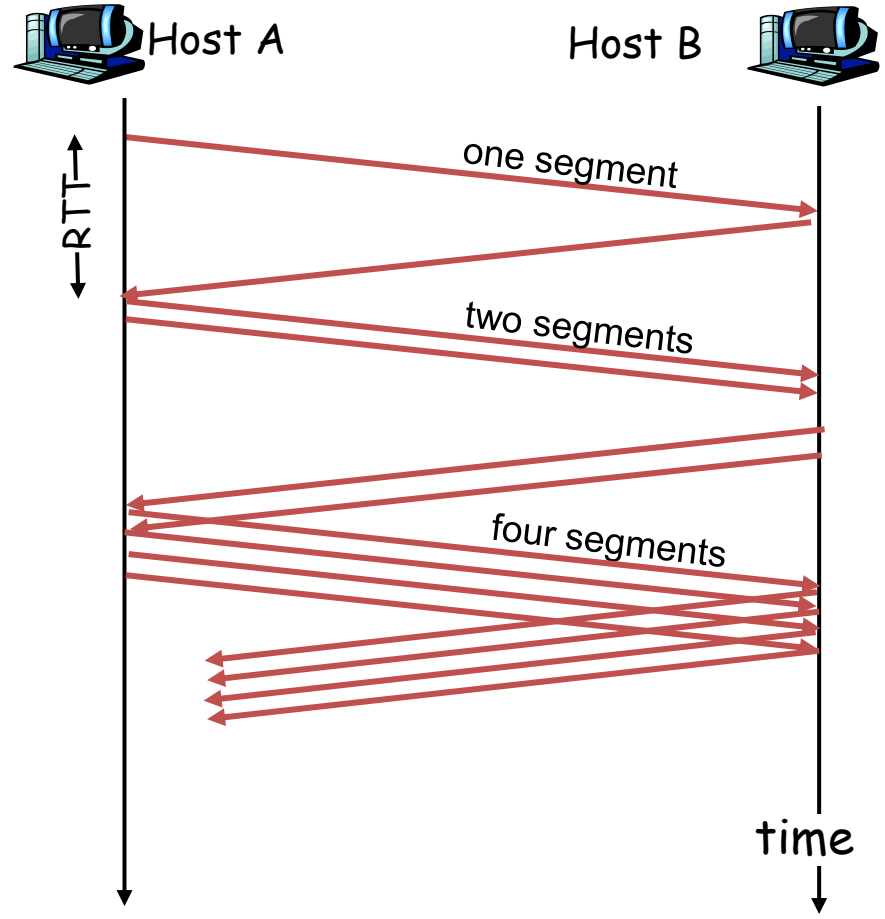
IMPROVEMENT IN TCP PERFORMANCE

Improvement in TCP Performance

- Traditional TCP
  - Slow Start
  - Fast Retransmit and recovery
- Popular TCP Congestion Control Algorithms
  - TCP Tahoe
  - TCP Reno
  - TCP SACK
  - TCP Vegas
- TCP in Mobile Networks

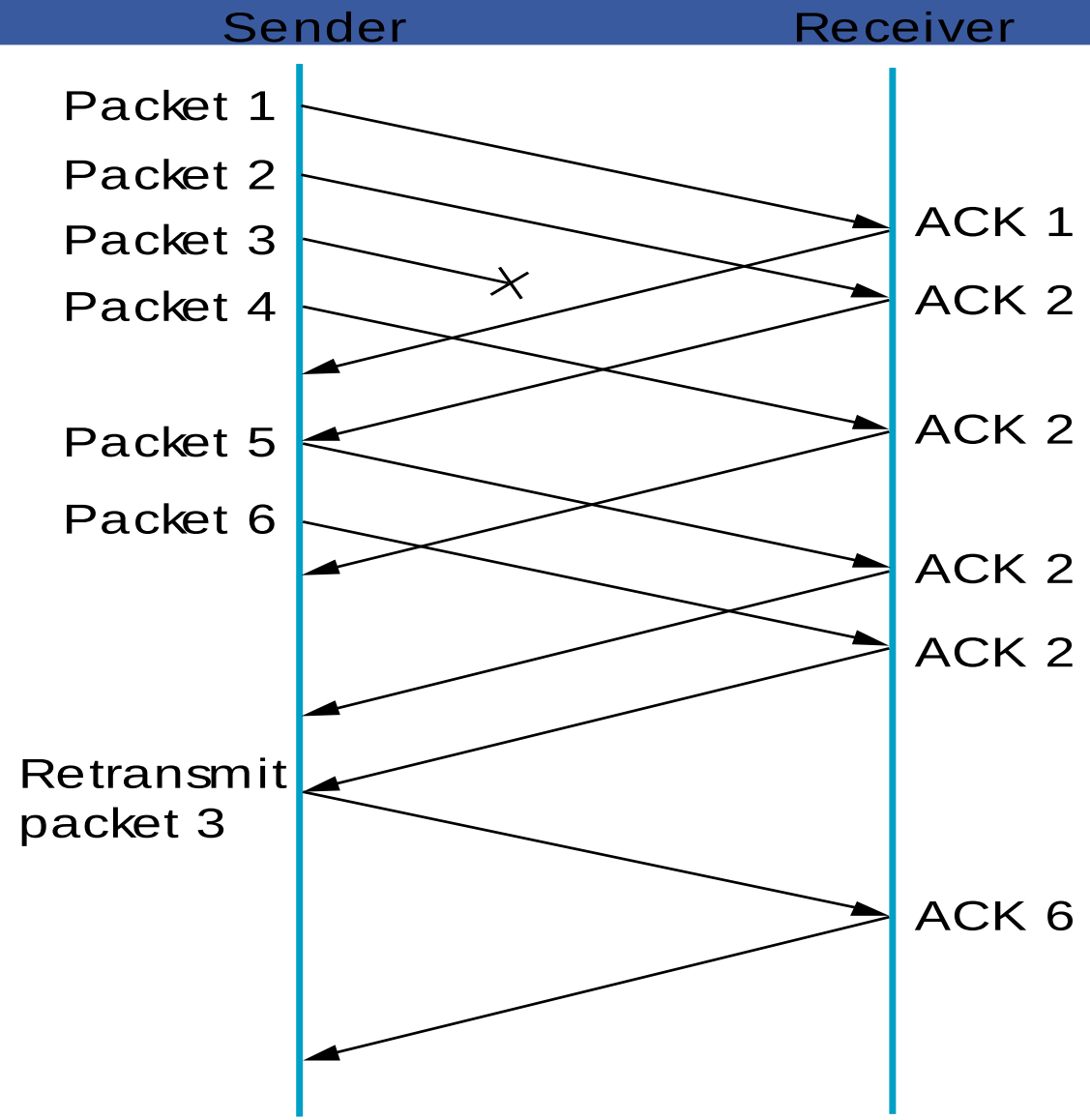
TCP Slow-Start algorithm

- Sender calculates a congestion window for a receiver
- Start with a congestion window size equal to one segment
- Exponential increase of the congestion window up to the congestion threshold, then linear increase
- Missing acknowledgement causes the reduction of the congestion threshold to one half of the current congestion window
- Congestion window starts again with one segment



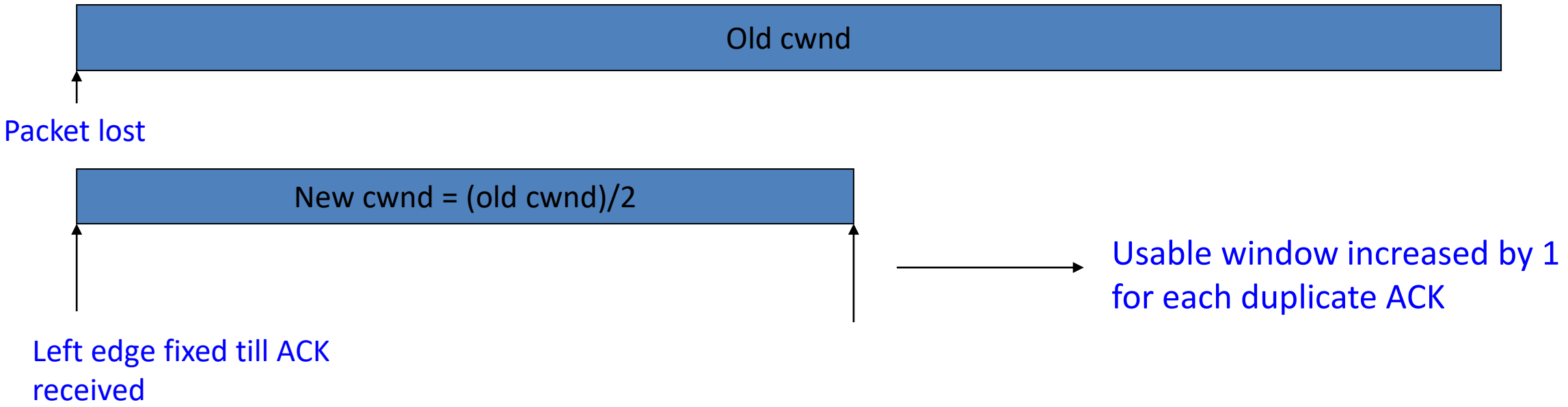
Fast Retransmit

- Coarse timeouts remained a problem, and Fast retransmit was added with TCP.
- Since the receiver responds every time a packet arrives, this implies the sender will see duplicate ACKs.
- Use duplicate ACKs to signal lost packet.
- Upon receipt of *three duplicate ACKs,* the TCP Sender retransmits the lost packet.



Fast Recovery

- Fast recovery was added with TCP.
- When fast retransmit detects three duplicate ACKs, start the recovery process from congestion
- After Fast Retransmit, half the cwnd and commence *recovery from this point using linear additive increase.*



Popular TCP Congestion Control Algs

- Comparison
  - Tahoe: Slow start, fast retransmit
  - Reno: Tahoe + fast recovery
  - New-Reno: Reno with modified fast recovery
  - SACK: Reno + selective ACKs
  - Vegas: Modified Slow start, and retransmission

TCP Tahoe

Packet loss is identified in 2 ways:

1. Time out
2. Dup acks

- Slow start, fast retransmit
- Fast retransmit improves channel utilization
- Two slow start situations:
  - At the very beginning of a connection **{cold start}**.
  - When the connection goes dead waiting for a timeout to occur (i.e, the advertized window goes to zero!)

TCP Tahoe

- Coarse timeouts remained a problem, and Fast retransmit was added with **TCP Tahoe**.
- Since the receiver responds every time a packet arrives, this implies the sender will see duplicate ACKs.

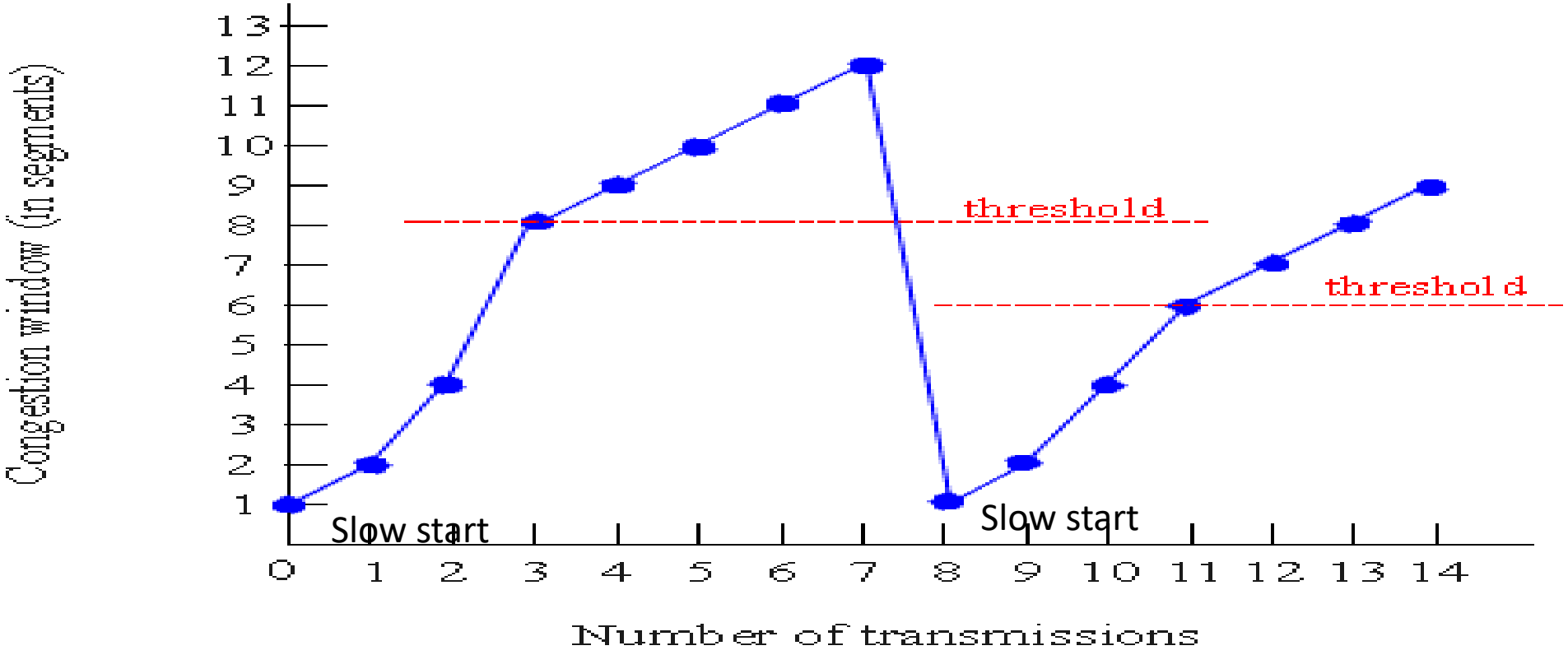
Basic Idea:: *use **duplicate ACKs** to signal lost packet.*

Fast Retransmit

Upon receipt of three duplicate ACKs, the TCP Sender retransmits the lost packet.

TCP Tahoe

- Limitations
  - Too aggressive
  - Returns to slow start on every congestion



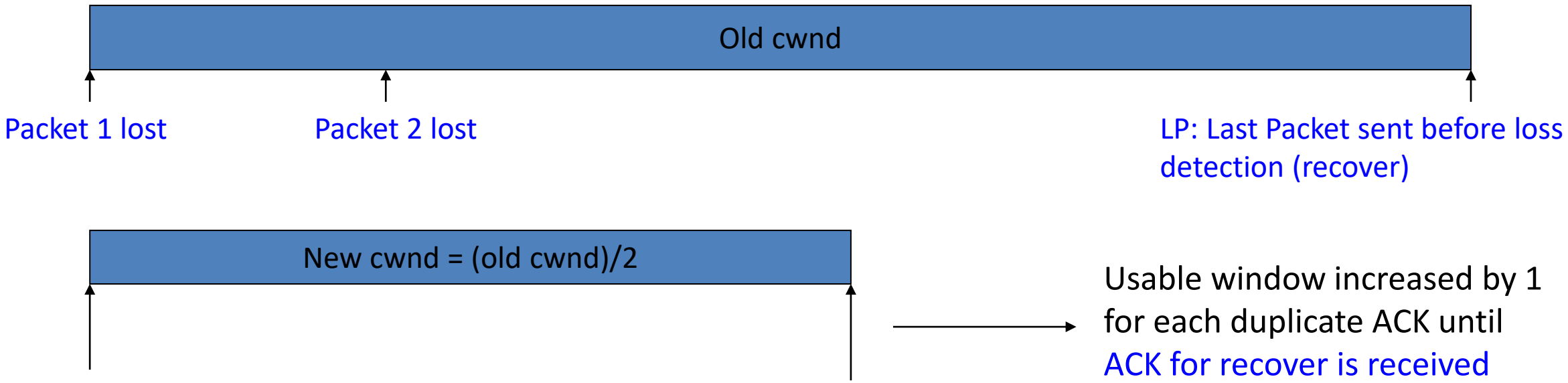
TCP Reno

- Tahoe + fast recovery
- After fast retransmit, reduce cwnd by half, and continue sending segments at this reduced level. (Instead of moving to slow start)
- Limitations
  - Has to wait for 3 Dup-ACKs
  - Inefficient incase of multiple losses

TCP New-Reno

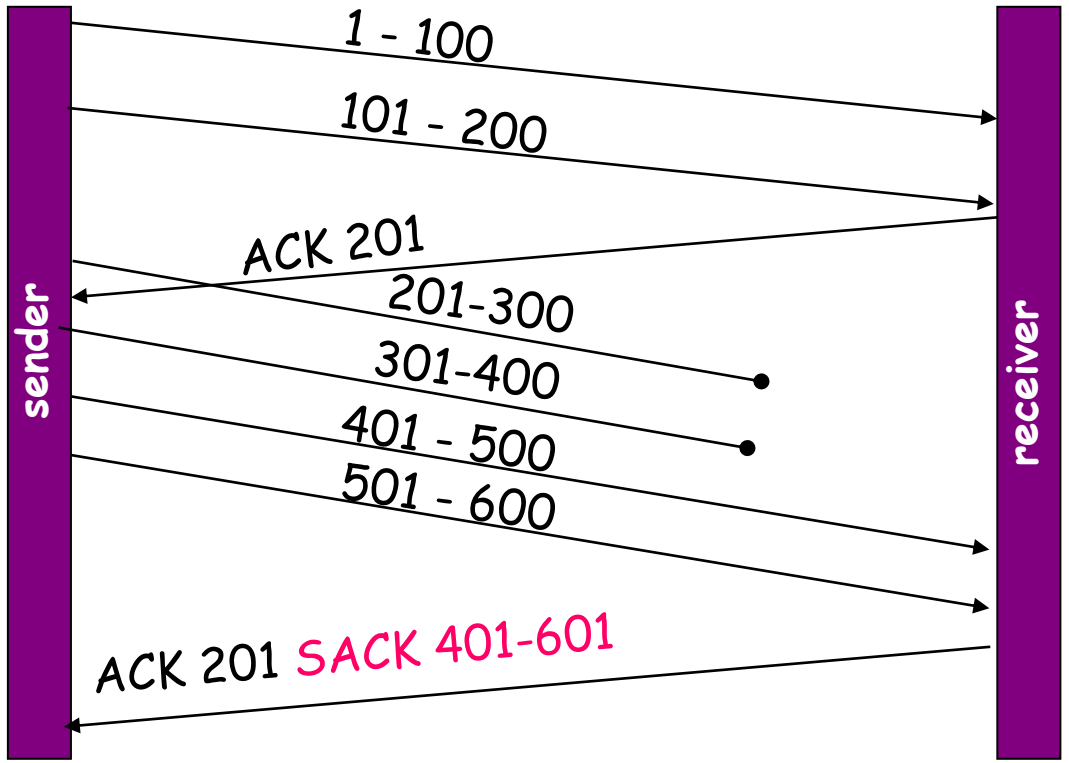
- Reno with modified fast recovery
- New-Reno continues with fast recovery if a *partial ACK* is received
- When duplicate ACKs trigger a retransmission for a lost packet, remember the highest packet sent from window in recover.
- In sender side, upon receiving an ACK,
  - if ACK < recover => partial ACK (for old segment sent)
  - If ACK ≥ recover => new ACK (for previous segment sent)
- Partial ACK implies another lost packet: retransmit next packet, inflate window and stay in fast recovery.
- New ACK implies fast recovery is over: starting from 0.5 x cwnd proceed with congestion avoidance (linear increase).
- New Reno recovers from n losses in n round trips.

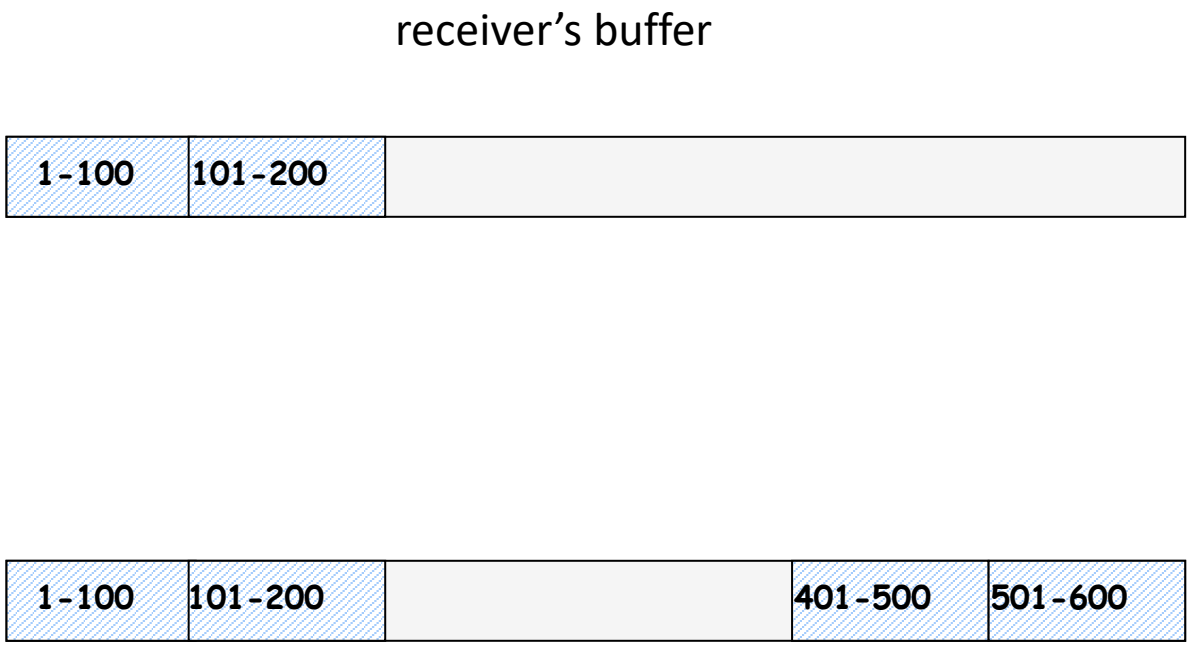
TCP New-Reno


Old cwnd
Packet 1 lost
Packet 2 lost
LP: Last Packet sent before loss detection (recover)
New cwnd = (old cwnd)/2
Usable window increased by 1 for each duplicate ACK until ACK for recover is received

TCP SACK

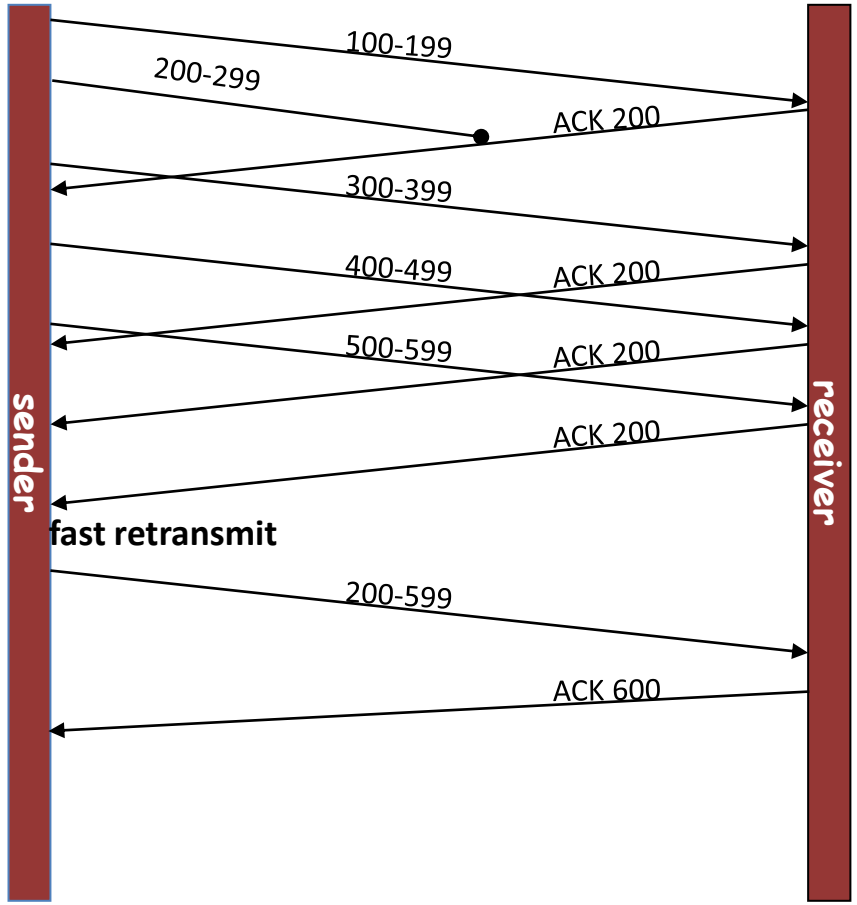
- Reno + selective ACKs



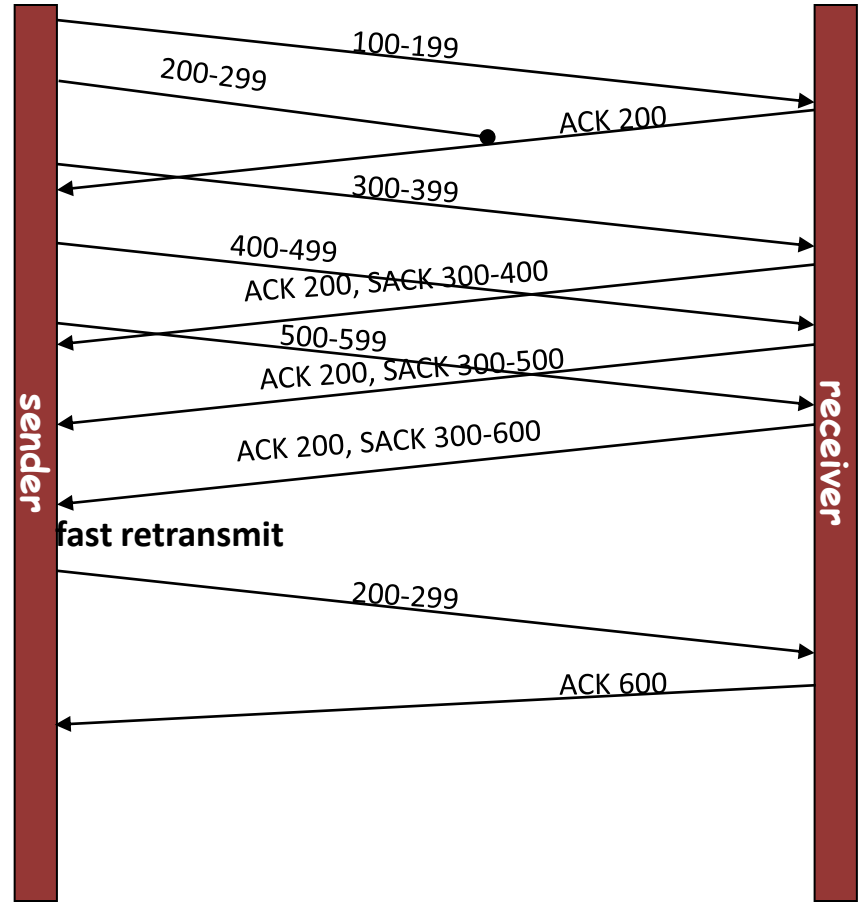


TCP SACK

TCP without SACK



TCP with SACK



TCP Vegas

- The only way Tahoe, Reno and New Reno can detect congestion is by creating congestion!
  - They carefully probe for congestion by slowly increasing their sending rate.
  - When they find (create), congestion, they cut sending rate at least in half!

TCP Vegas

- There are three ways proposed in Vegas to increase delivery throughput and decrease packet loss.
  - Modified Slow-Start Mechanism
    - Cwnd is allowed exponential growth only every other RTT. (doubles the size of cwnd every 2 RTT time while there are no losses).
  - New Congestion Avoidance Mechanism
    - Control the size of cwnd by observing the variation of RTT
    - Congestion in network (high RTT)
  - New Retransmission Mechanism
    - Sender does retransmission after a dupACK received, if RTT estimate > timeout (without waiting for 3 dupACKs)

Summary

- Transmission Control Protocol(TCP)
  - Basic concepts of TCP
  - TCP header
- Adaptation of TCP window
  - Silly window syndrome
  - Adaptive retransmision
- Improvements in TCP (Wired TCP)
  - TCP Tahoe
  - TCP Reno
  - TCP New-Reno
  - TCP SACK
  - TCP Vegas

Test your understanding

- Why do congestion occur in a network?

- How does slow start help improve the performance of TCP?

References

- Prasant Kumar Pattnaik, Rajib Mall, "Fundamentals of Mobile Computing", PHI Learning Pvt. Ltd, New Delhi – 2012.
- Jochen H. Schller, "Mobile Communications", Second Edition, Pearson Education, New Delhi, 2007.
- Behrouz A. Forouzan, "Data communication and Networking", Fourth Edition,Tata McGraw – Hill, 2011.