Transport Protocols

Kameswari Chebrolu
Dept. of Electrical Engineering, IIT Kanpur

End-to-End Protocols

- Convert host-to-host packet delivery service into a process-to-process communication channel
  - Demultiplexing: Multiple applications can share the network
- End points identified by ports
  - Ports are not interpreted globally
  - servers have well defined ports (look at /etc/services)

Application Layer Expectations

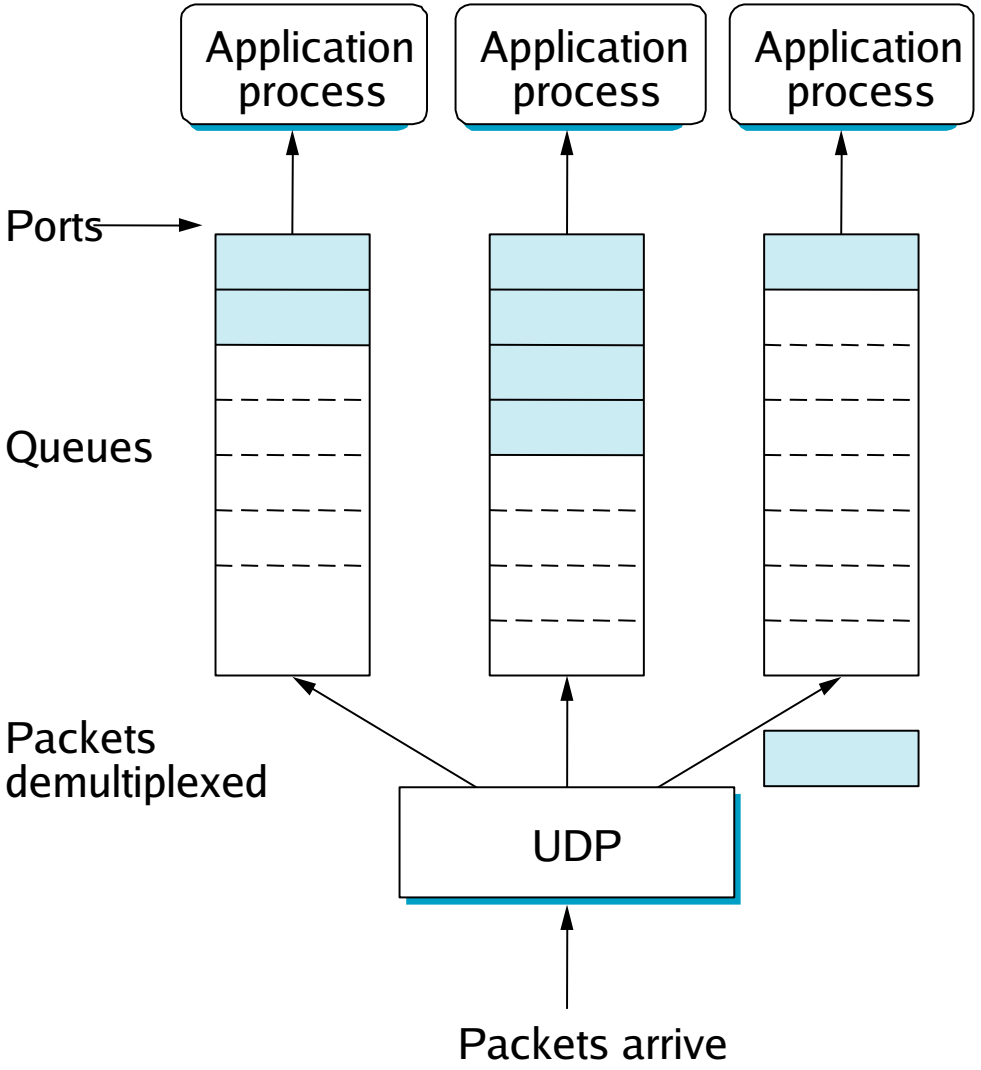
- Guaranteed message delivery
- Ordered delivery
- No duplication
- Support arbitrarily large messages
- Synchronization between the sender and receiver
- Support flow control
- Support demultiplexing

Limitations of Networks

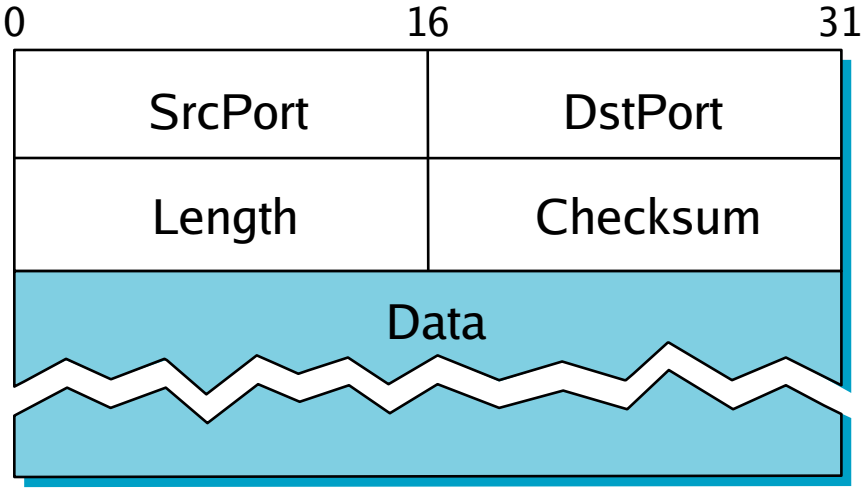
- Packet Losses
- Re-ordering
- Duplicate copies
- Limit on maximum message size
- Long delays

User Datagram Protocol (UDP)

Demultiplexing



UDP Header

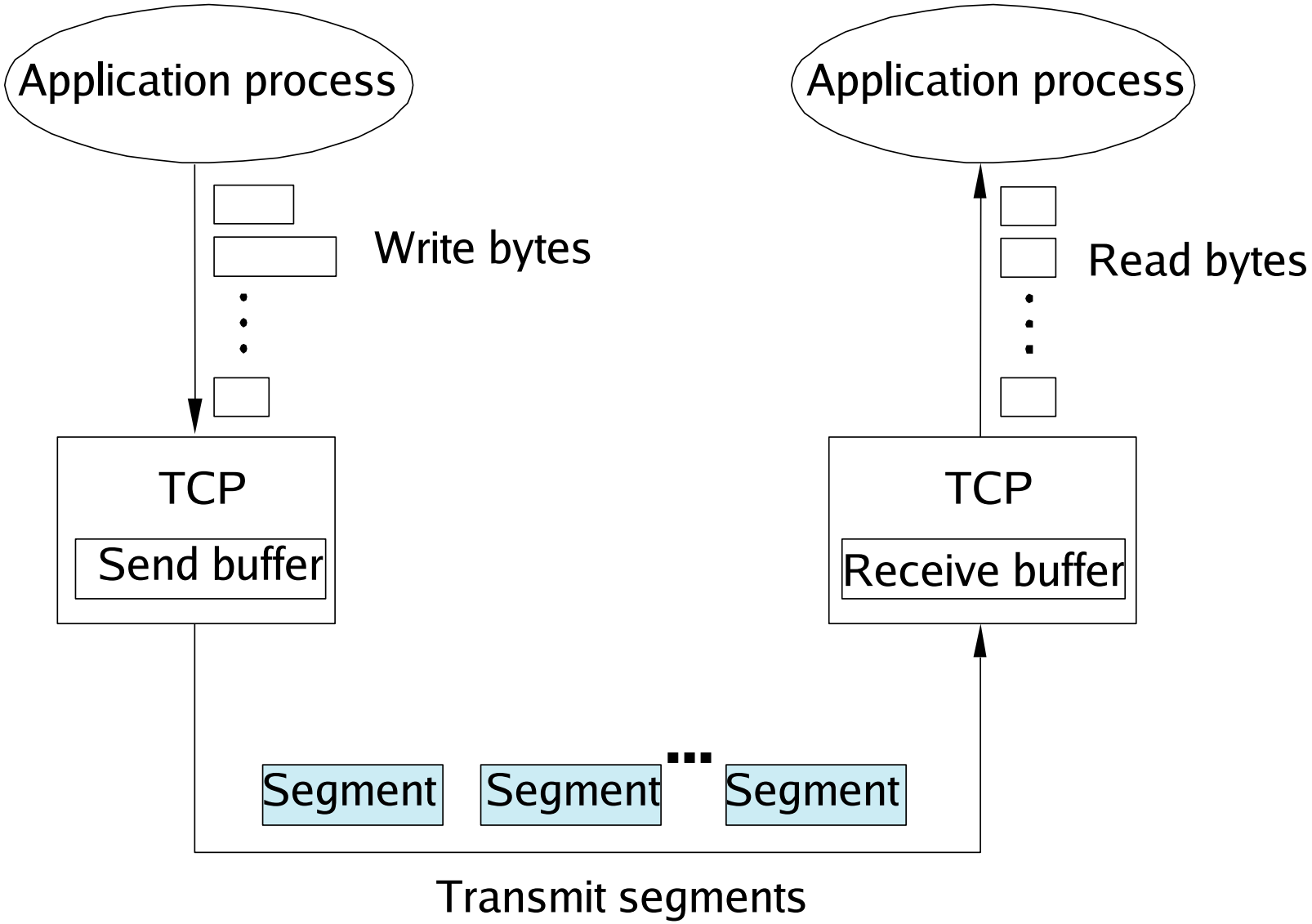


Computes checksum over UDP header, message body and pseudo-header

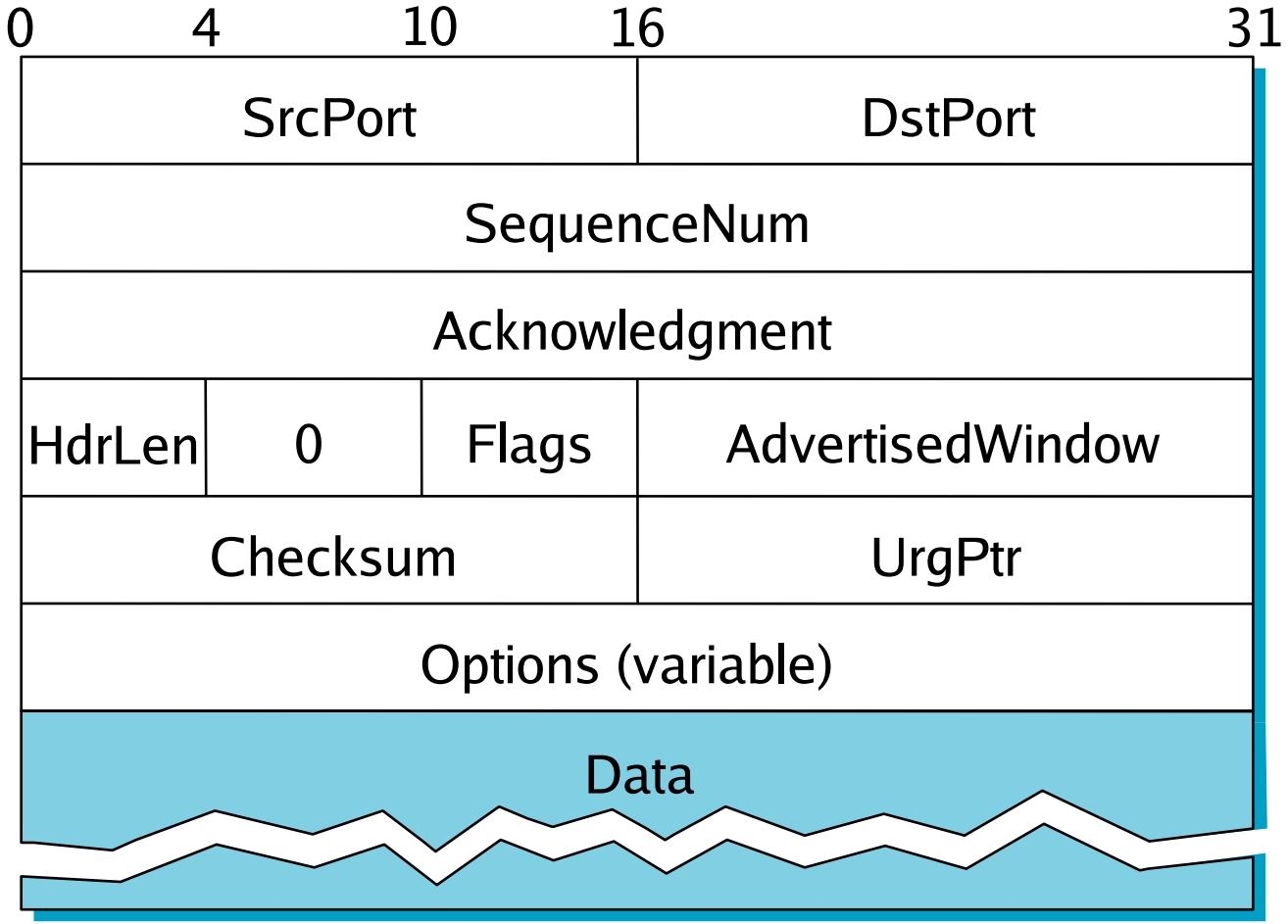
Transmission Control Protocol (TCP)

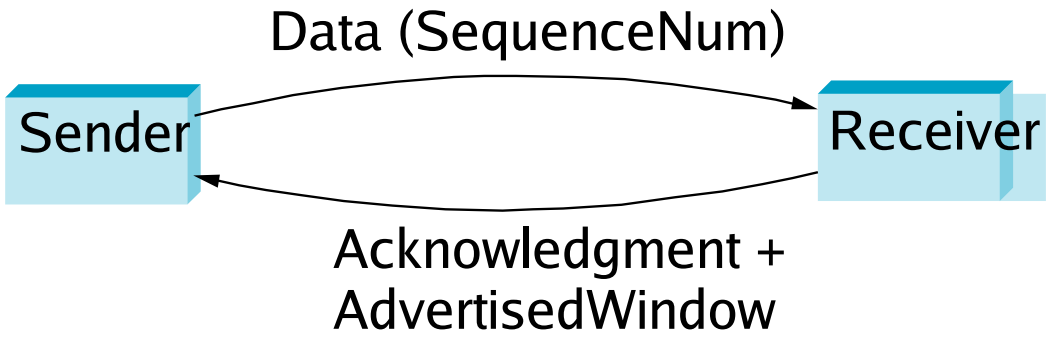
- Connection oriented
  - Maintains state to provide reliable service
- Byte-stream oriented
  - Handles byte streams instead of messages
- Full Duplex
  - Supports flow of data in each direction
- Flow-control
  - Prevents sender from overrunning the receiver
- Congestion-control
  - Prevents sender from overloading the network

TCP Cont...


Application process
Application process
Write bytes
Read bytes
TCP
Send buffer
TCP
Receive buffer
Segment
Segment
Segment
Transmit segments

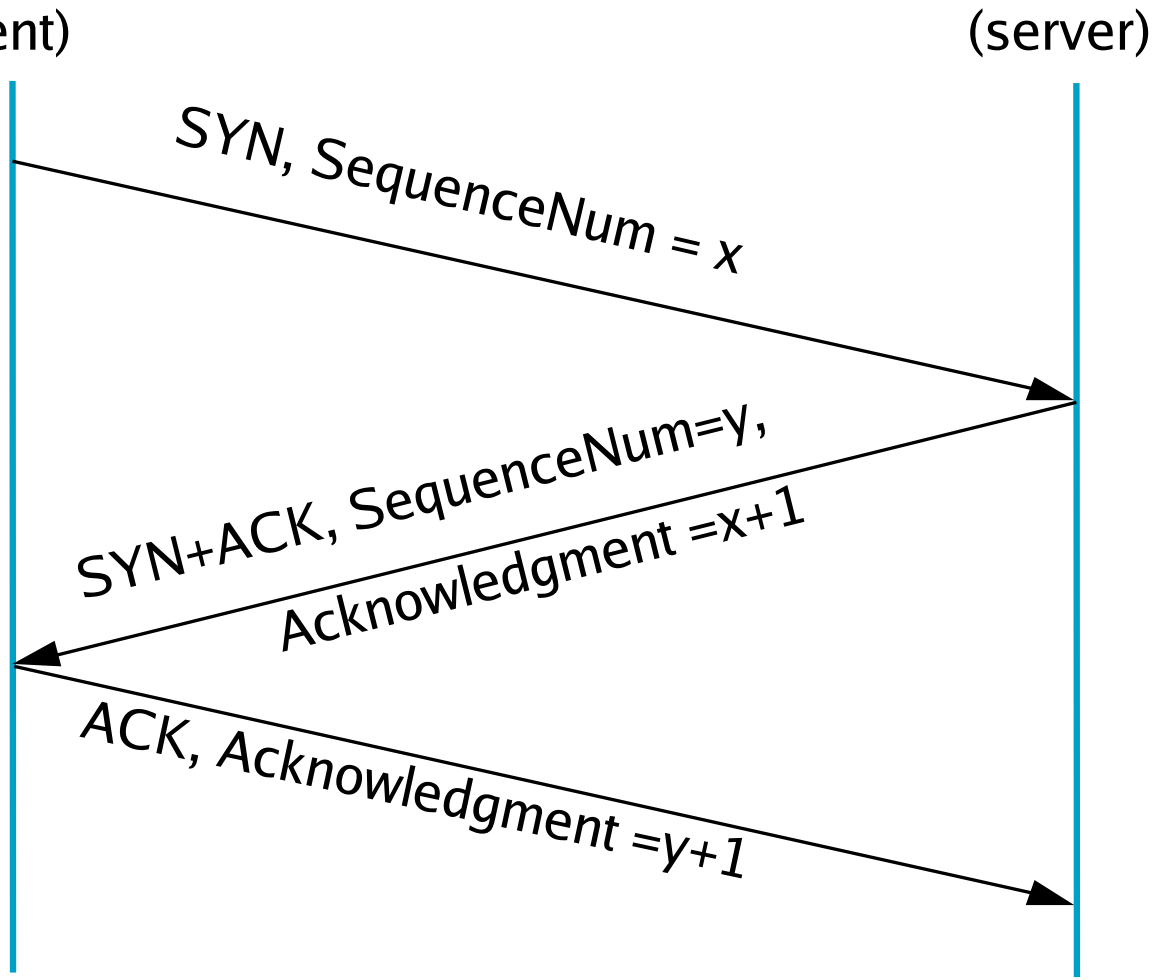
TCP Header Format



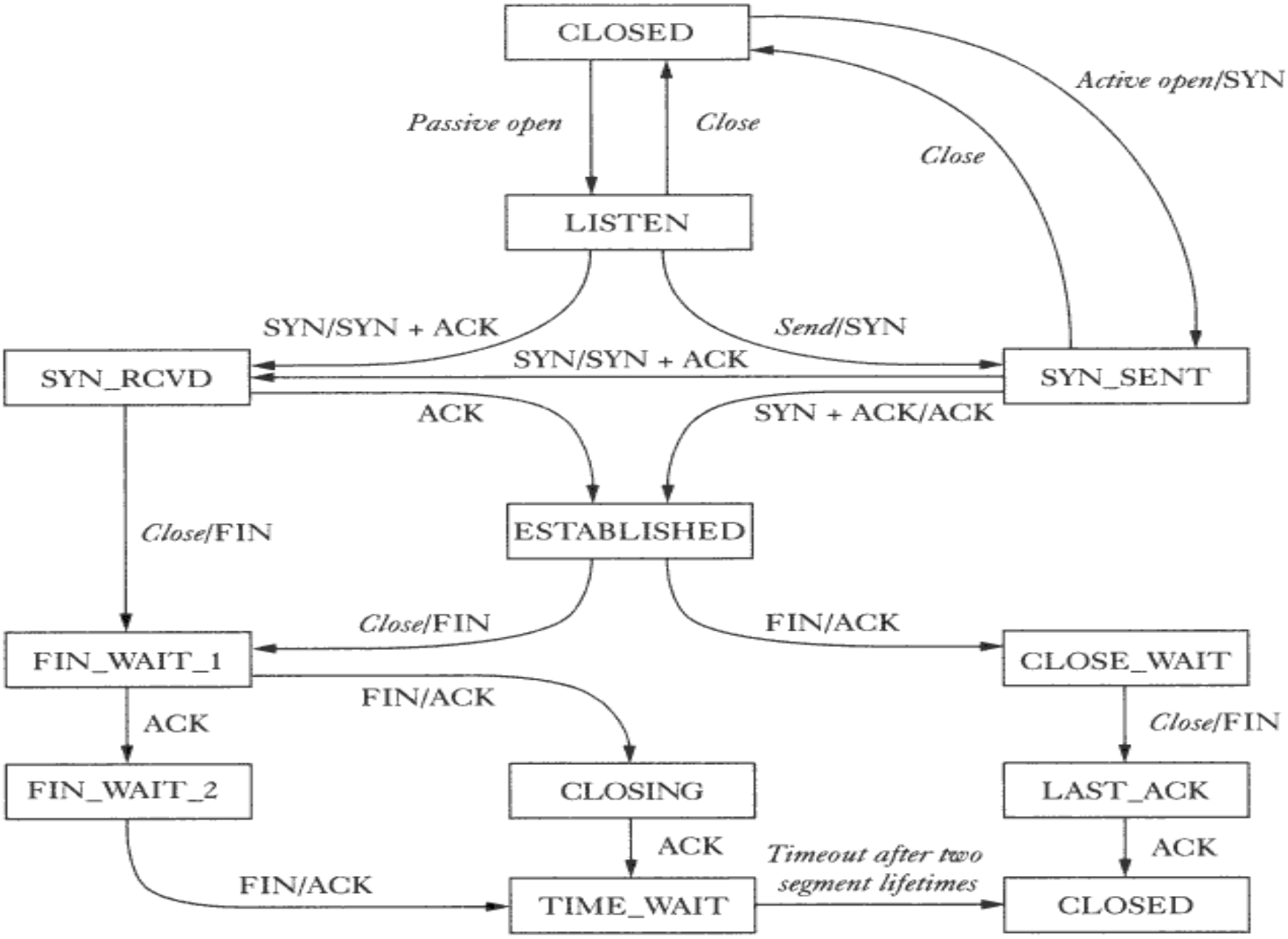


Connection Establishment

Active participant



State Transition Diagram



Sliding Window: Data Link vs Transport

P2P: End points can be engineered to support the link

TCP: Any kind of computer can be connected to the Internet

- Need mechanism for each side to learn other side's resources (e.g. buffer space) -- Flow control

P2P: Not possible to unknowingly congest the link

TCP: No idea what links will be traversed, network capacity can dynamically vary due to competing traffic

- Need mechanism to alter sending rate in response to network congestion – Congestion control

Sliding Window: Data Link vs Transport

P2P: Dedicated Link -- Physical Link connects the same two computers

TCP: Connects two processes on any two machines in the Internet

- Needs explicit connection establishment phase to exchange state

P2P: Fixed round trip transmission time (RTT)

TCP: Potentially different and widely varying RTTs

- Timeout mechanism has to be adaptive

P2P: No Reordering

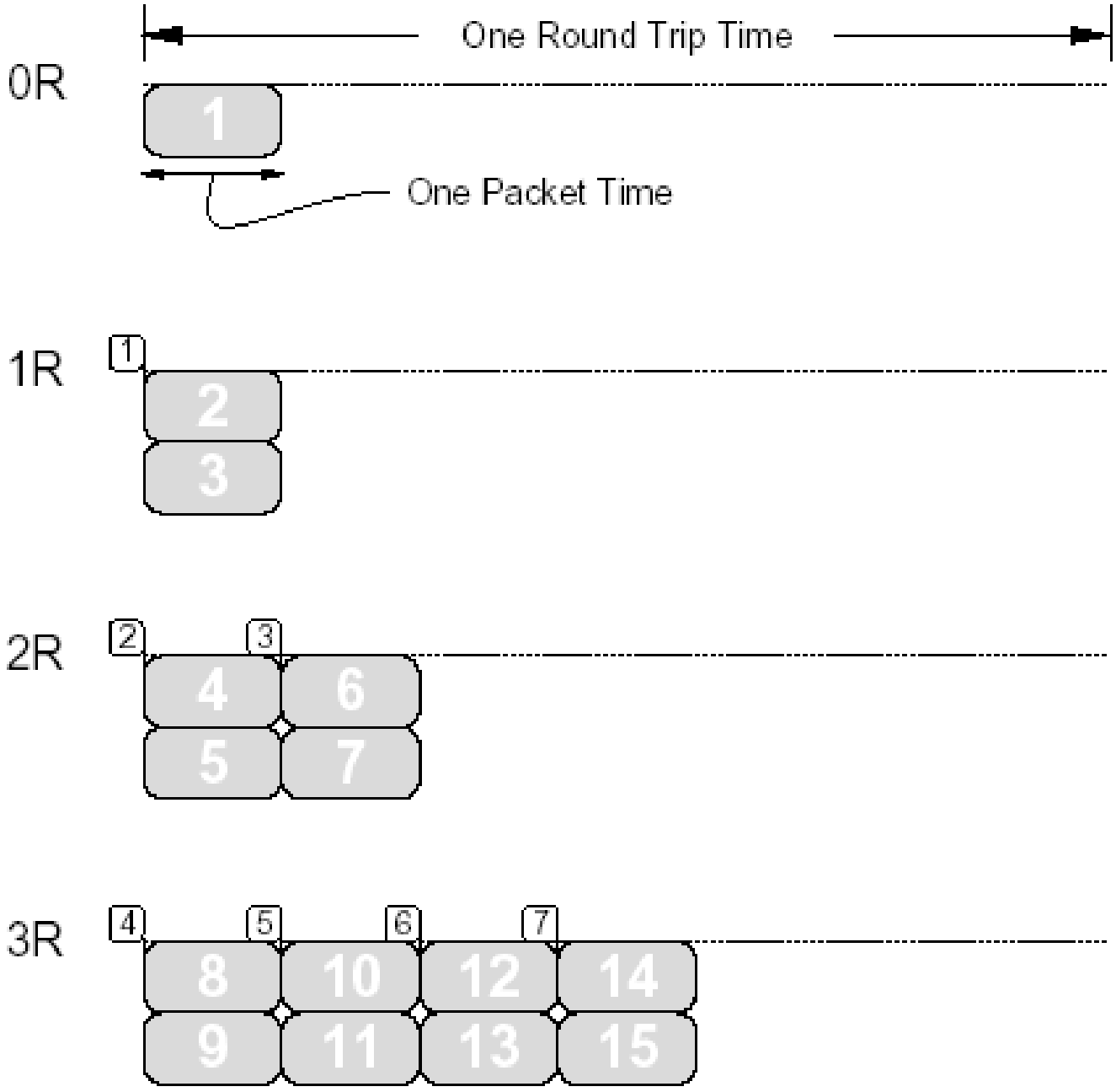
TCP: Scope for reordering due to arbitrary long delays

- Need to be robust against old packets showing up suddenly

Slow Start

- Add a variable cwnd (congestion window)
- At start, set cwnd=1
- On each ack for new data, increase cwnd by 1
- When sending, send the minimum of receiver's advertised window or cwnd

Figure 2: The Chronology of a Slow-start



Congestion Avoidance

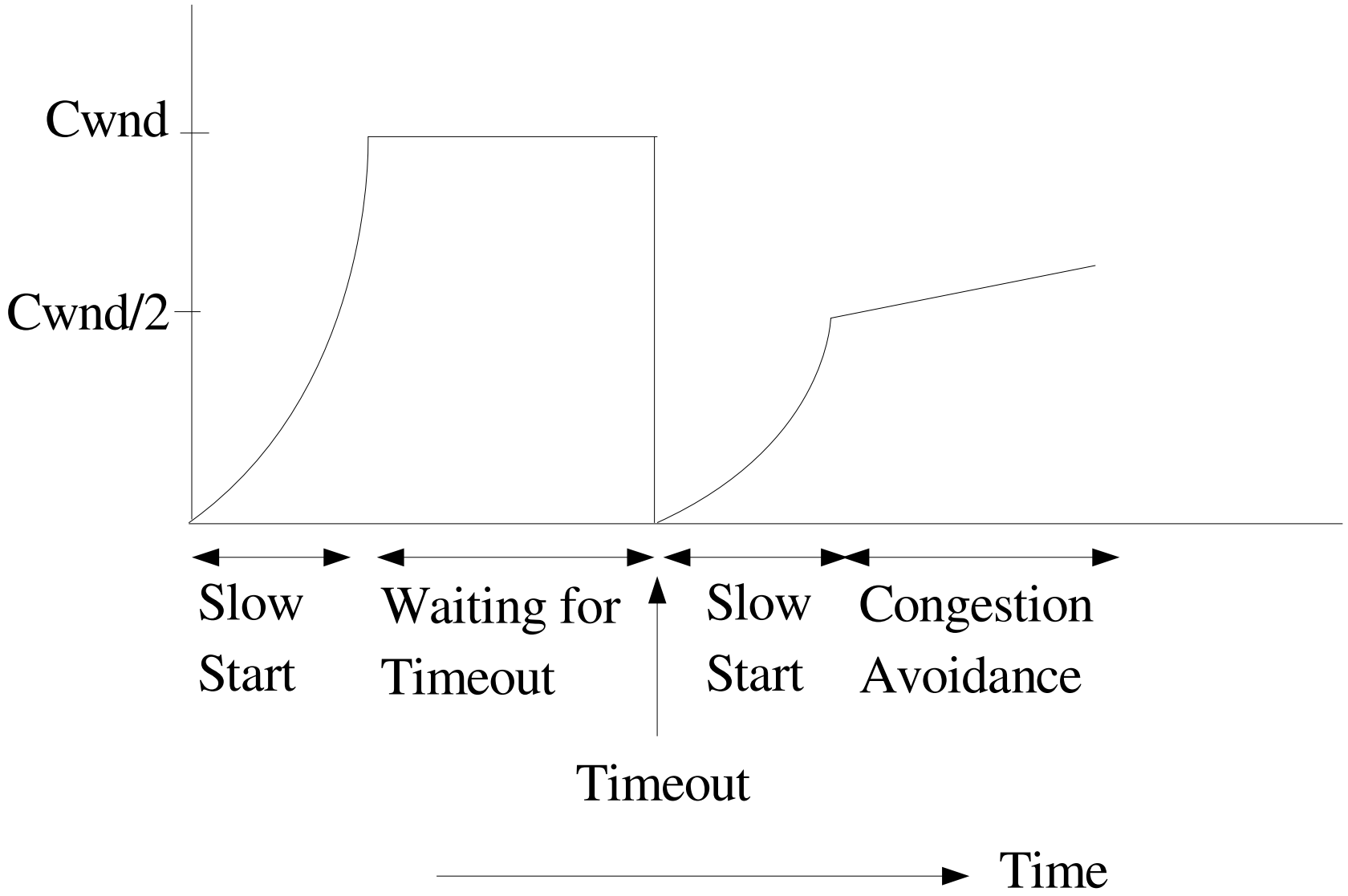
(Additive Increase, Multiplicative Decrease)

- On detecting congestion, set cwnd to half the window size (multiplicative decrease)
- On each ack of new data, increase cwnd by 1/cwnd (additive increase)

Combining Slow Start and Congestion Avoidance

- Two variables cwnd and ssthresh
- On time out, set ssthresh = cwnd/2; cwnd = 1
- When new data is acked,
  - If (cwnd < ssthresh) cwnd += 1;
  - Else cwnd += 1/cwnd;

Congestion Window vs Time

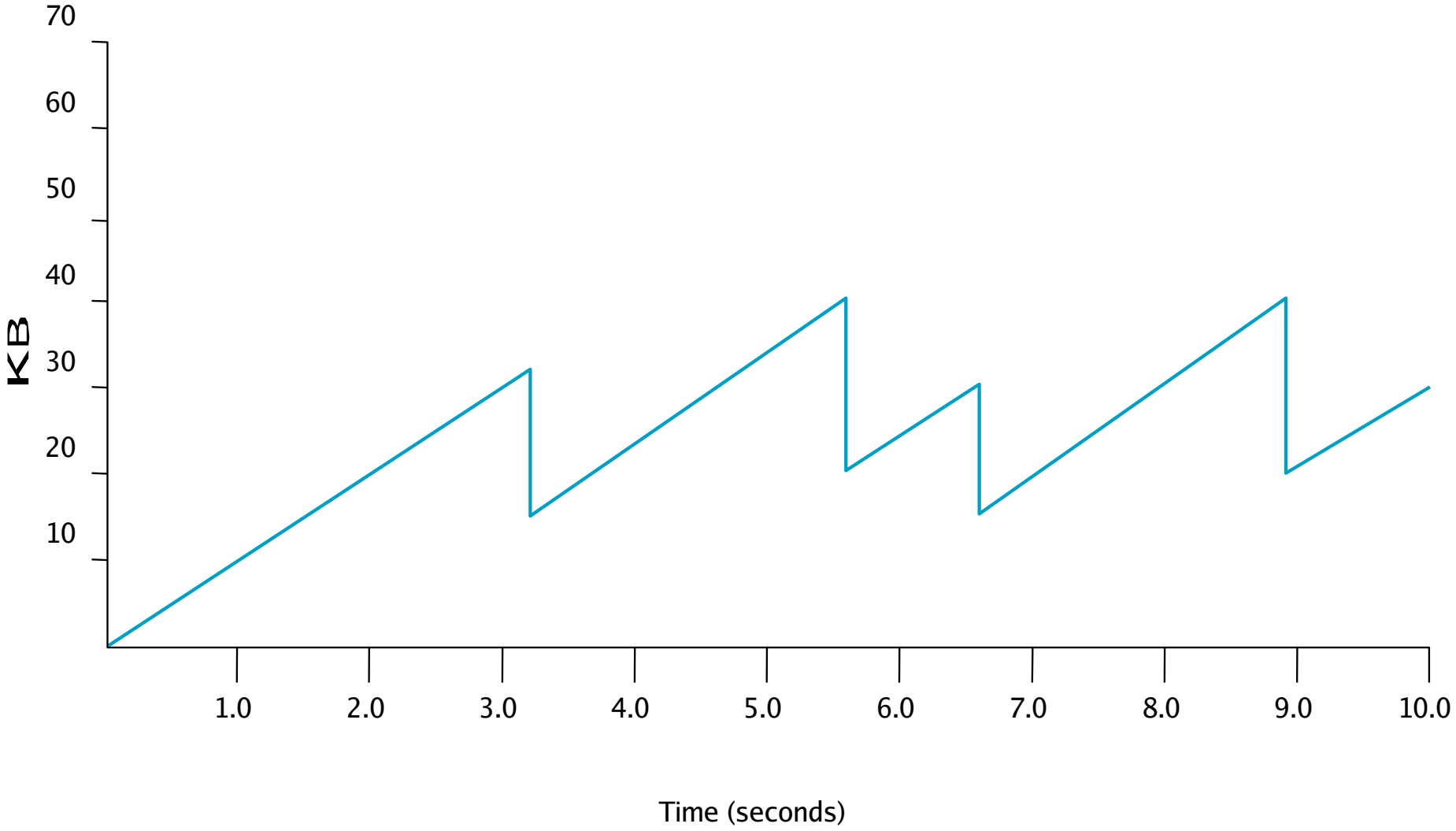


Fast Retransmit & Fast Recovery

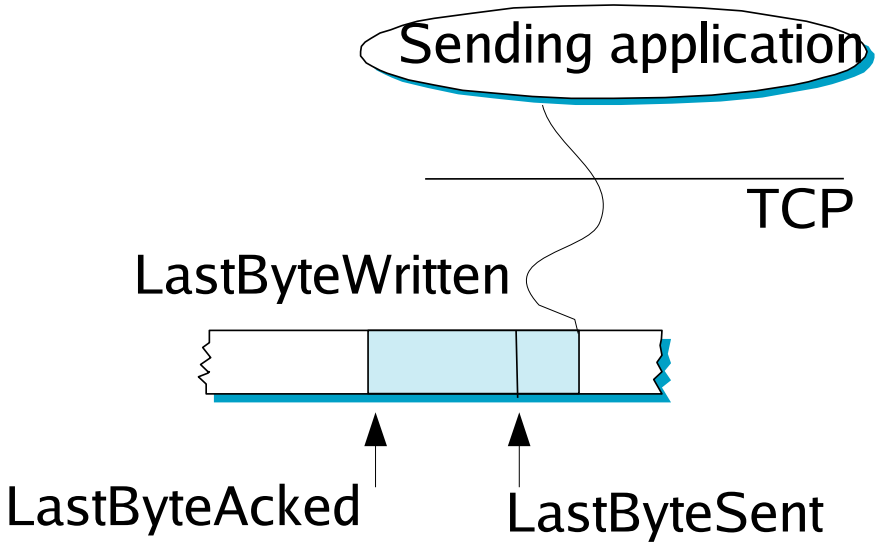
- Fast Retransmit: Retransmit packet at sender after 3 duplicate acks
- Fast Recovery
  - On 3rd dupack, retransmit packet, ssthresh = min (cwnd/2,2); cwnd = ssthresh+3
  - Another dupack, cwnd = cwnd +1; transmit packet if allowed by cwnd
  - On ack acknowledging new data, cwnd = ssthresh, invoke congestion avoidance (linear increase in cwnd now on)

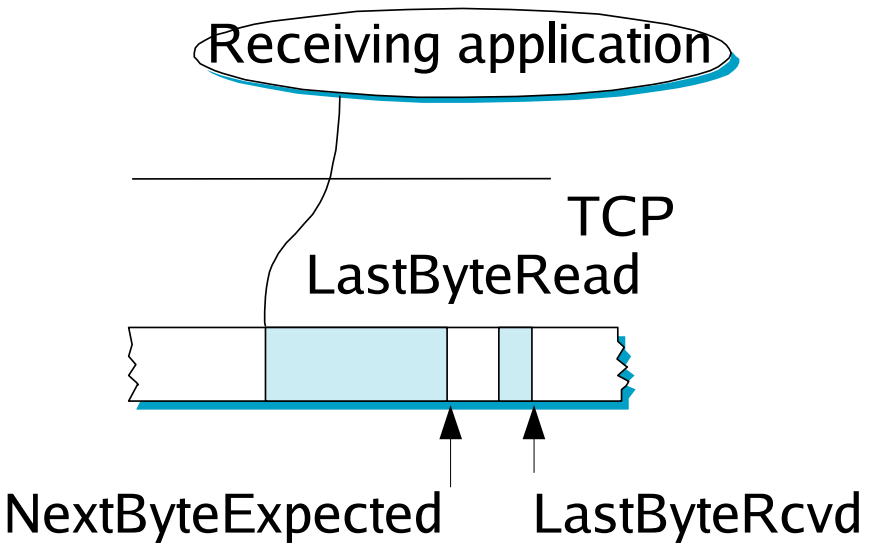
Saw Tooth Pattern

(With fast retransmit and recovery)



Sliding Window Recap





Sending Side:

- LastByteAcked <= LastByteSent
- LastByteSent <= LastByteWritten
- Buffer bytes between LastByteAcked and LastByteWritten

Receiving Side:

- LastByteRead <= NextByteExpected
- NextByteExpected <= LastByteRcvd+1
- Buffer bytes between LastByteRead and LastByteRcvd

Flow & Congestion Control

- Buffers are of finite size
  - MaxSendBuffer and MaxRcvBuffer
- Receiving side:
  - LastByteRcvd – LastByteRead <= MaxRcvBuffer
  - AdvertisedWindow = MaxRcvBuffer – ((NextByteExpected – 1) – LastByteRead)
- Sending side:
  - MaxWindow = min (cwnd, AdvertisedWindow)
  - EffectiveWindow = MaxWindow – (LastByteSent – LastByteAcked)
  - LastByteWritten – LastByteAcked <= MaxSendBuffer
  - Persist when AdvertisedWindow is zero

RTT Estimation: Original Algorithm

- Measure SampleRTT for sequence/ack combo
- EstimatedRTT = a*EstimatedRTT + (1-a)*SampleRTT
  - *a* is between 0.8-0.9
  - small *a* heavily influenced by temporary fluctuations
  - large *a* not quick to adapt to real changes
- Timeout = 2 * EstimatedRTT

Jacobson/Karels Algorithm

- Incorrect estimation of RTT worsens congestion
- Algorithm takes into account variance of RTTs
  - If variance is small, EstimatedRTT can be trusted
  - If variance is large, timeout should not depend heavily on EstimatedRTT

Jacobson/Karels Algorithm Cont..

- Difference = SampleRTT - EstimatedRTT
- EstimatedRTT = EstimatedRTT + (d * Difference)
- Deviation = Deviation + d (|Difference| - Deviation)), where d ~ 0.125
- Timeout = u * EstimatedRTT + q * Deviation, where u = 1 and q = 4
- Exponential RTO backoff

.

Protection Against Wraparound

- Wraparound occurs because sequence number field is finite
  - 32 bit sequence number space
- Solution: Use time stamp option
- Maximum Segment Lifetime (MSL) is 120 sec

Bandwidth	Time until Wraparound
T1 (1.5Mbps)	6.6 hrs
Ethernet (10Mbps)	57 minutes
T3 (45 Mbps)	13 minutes
FDDI (100Mbps)	6 minutes
STS-3 (155Mbps)	4 minutes
STS-12 (622Mbps)	55 seconds
STS-24 (1.2Gbps)	28 seconds

Summary

- Transport protocols essentially demultiplexing functionality
- Examples: UDP, TCP, RTP
- TCP is a reliable connection-oriented byte-stream protocol
  - Sliding window based
  - Provides flow and congestion control

Must Reads

- D. Clark, "The Design Philosophy of the DARPA Internet Protocols", SIGCOMM, Palo Alto, CA, Sept 1988, pp. 106-114
- J. Saltzer, D. Reed, and D. Clark, "End-to-end Arguments in System Design". ACM Transactions on Computer Systems (TOCS), Vol. 2, No. 4, 1984, pp. 195-206
- Van Jacobson, "Congestion Avoidance and Control", ACM SIGCOMM, 1988