Transport Layer: TCP and UDP

Raj Jain

Washington University in Saint Louis
Saint Louis, MO 63130
Jain@wustl.edu

Audio/Video recordings of this lecture are available on-line at:

http://www.cse.wustl.edu/~jain/cse473-16/

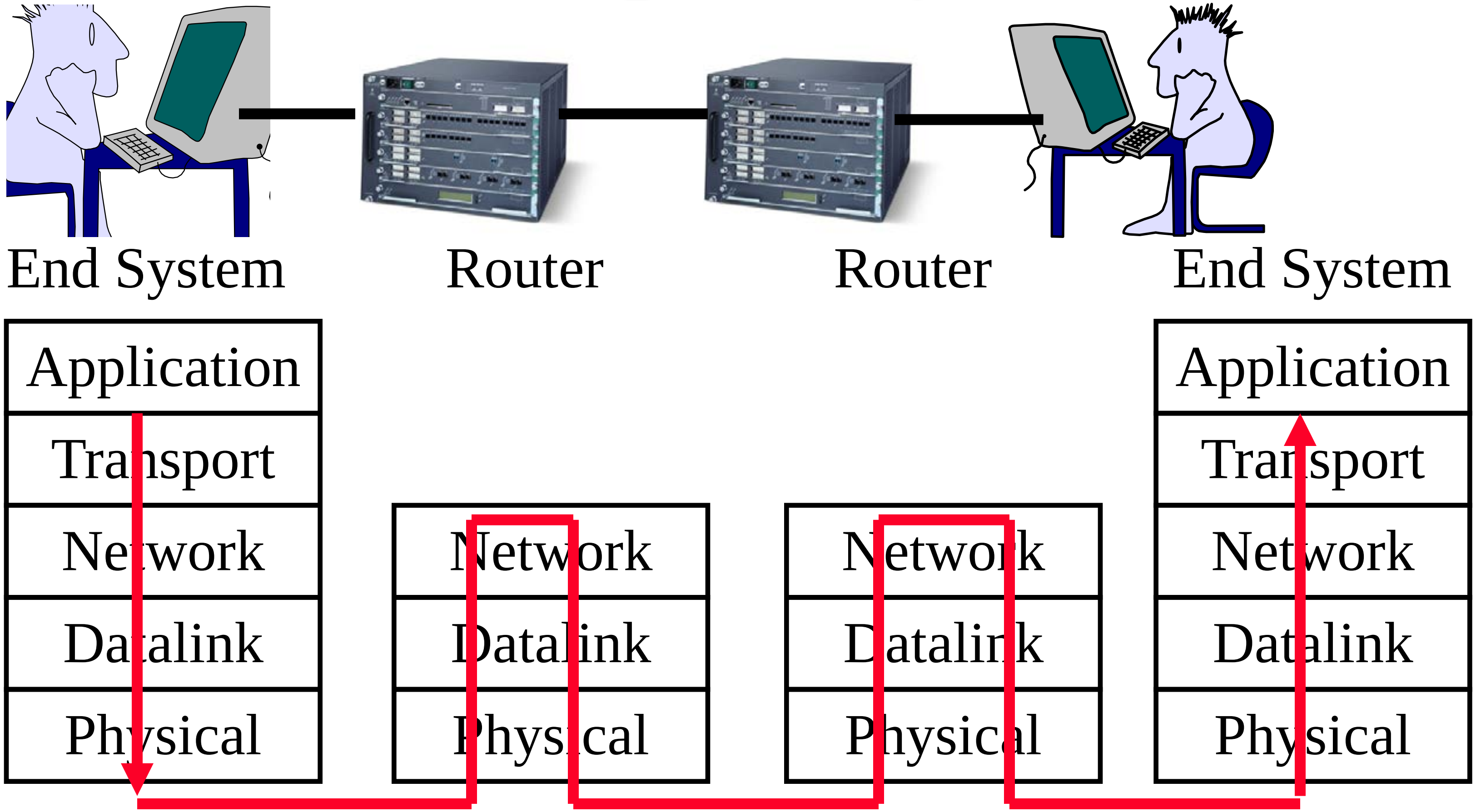
Overview

- Transport Layer Design Issues:
  - Multiplexing/Demultiplexing
  - Reliable Data Transfer
  - Flow control
  - Congestion control
- UDP
- TCP
  - Header format, connection management, checksum
  - Congestion Control
- **Note**: This class lecture is based on Chapter 3 of the textbook (Kurose and Ross) and the figures provided by the authors.

Transport Layer Design Issues

1. Transport Layer Functions
2. Multiplexing and Demultiplexing
3. Error Detection: Checksum
4. Flow Control
5. Efficiency Principle
6. Error Control: Retransmissions

Transport Layer



End System	Router	Router	End System
Application			Application
Transport			Transport
Network	Network	Network	Network
Datalink	Datalink	Datalink	Datalink
Physical	Physical	Physical	Physical

- Transport = End-to-End Services
  Services required at source and destination systems
  Not required on intermediate hops

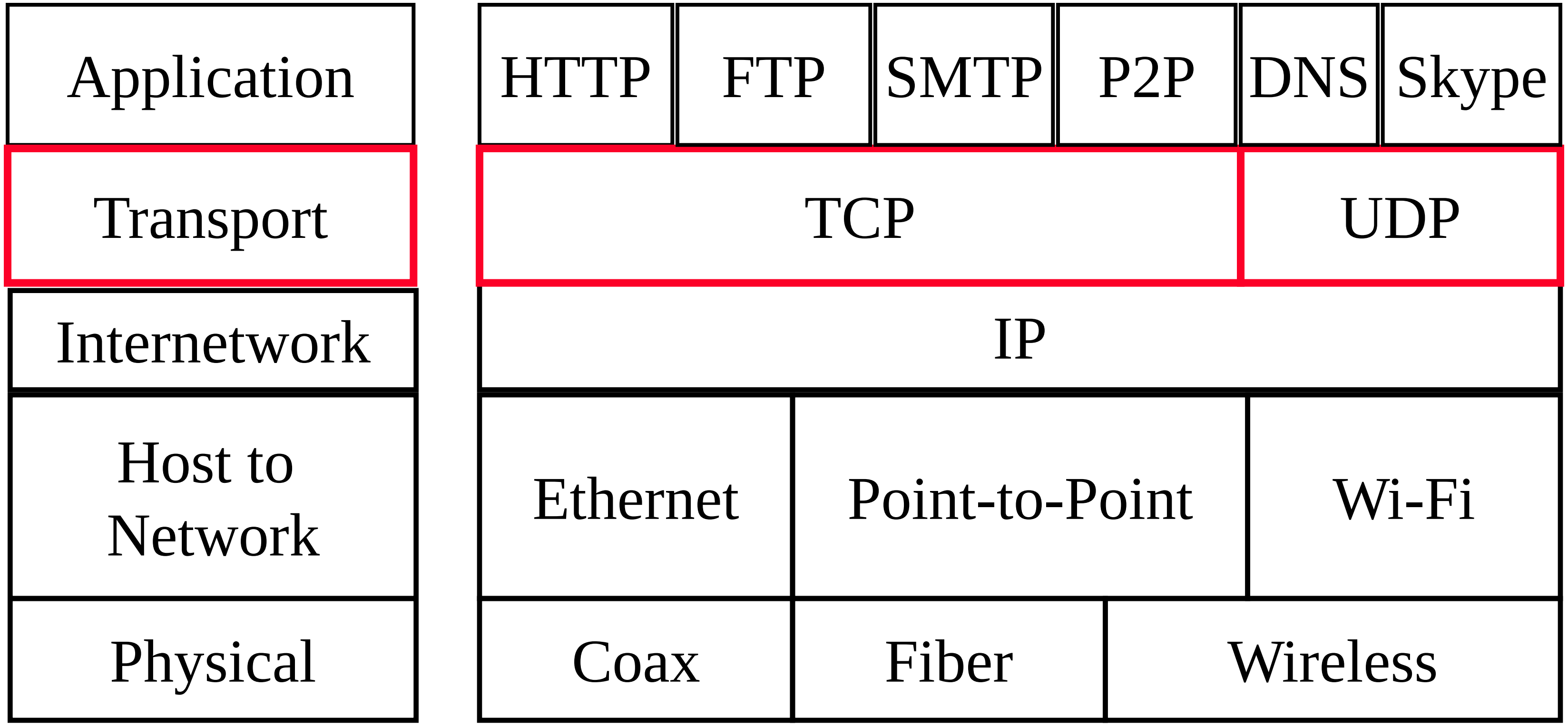
Transport Layer Functions

1. **Multiplexing and demultiplexing**: Among applications and processes at end systems
2. **Error detection**: Bit errors
3. **Loss detection**: Lost packets due to buffer overflow at intermediate systems (Sequence numbers and acks)
4. **Error/loss recovery**: Retransmissions
5. **Flow control**: Ensuring receiver has buffers
6. **Congestion Control**: Ensuring network has capacity

Not all transports provide all functions

Protocol Layers

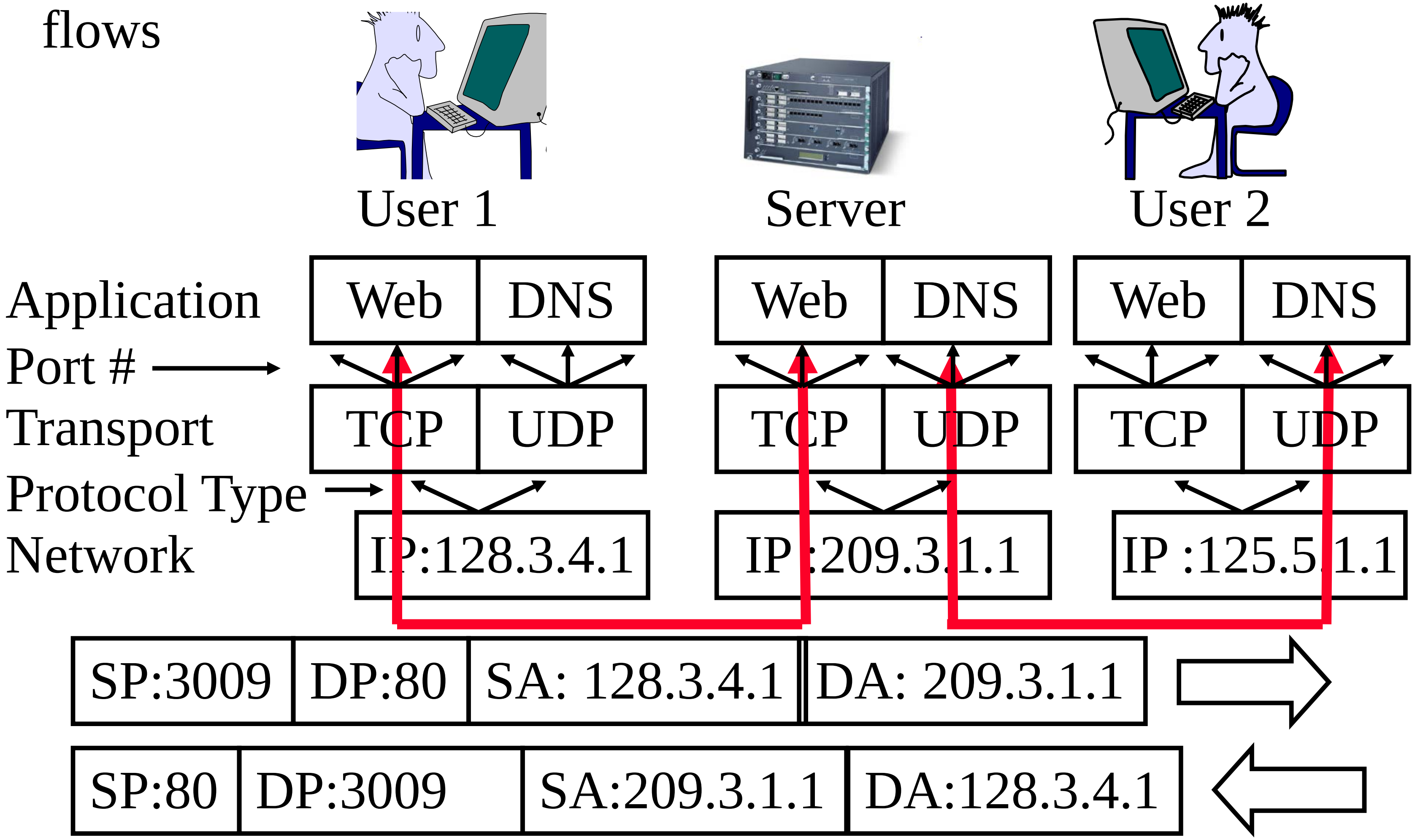
- Top-Down approach



Application	HTTP, FTP, SMTP, P2P, DNS, Skype
Transport	TCP, UDP
Internetwork	IP
Host to Network	Ethernet, Point-to-Point, Wi-Fi
Physical	Coax, Fiber, Wireless

Multiplexing and Demultiplexing

- Transport **Ports** and Network **addresses** are used to separate flows



User 1

Server

User 2

Application

Port #

Transport

Protocol Type

Network

User 1		Server		User 2	
Web	DNS	Web	DNS	Web	DNS
TCP	UDP	TCP	UDP	TCP	UDP
IP:128.3.4.1		IP :209.3.1.1		IP :125.5.1.1	

SP:3009	DP:80	SA: 128.3.4.1	DA: 209.3.1.1
SP:80	DP:3009	SA:209.3.1.1	DA:128.3.4.1

Ref: http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

User Datagram Protocol (UDP)

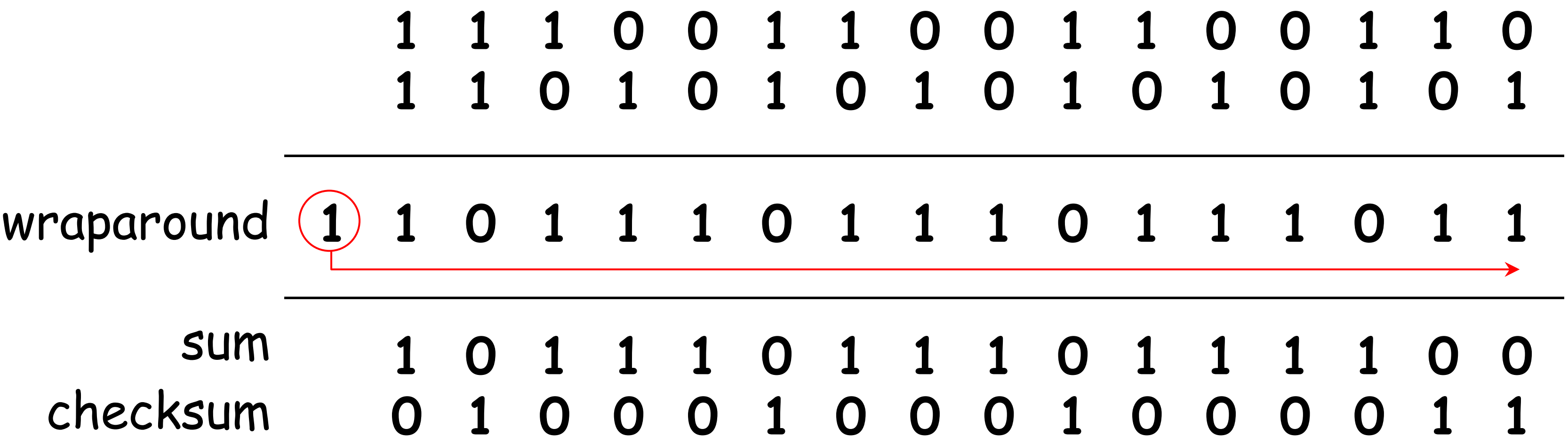
- Connectionless end-to-end service
- Provides multiplexing via ports
- Error detection (Checksum) optional. Applies to **pseudo-header** (same as TCP) and UDP segment. If not used, it is set to zero.
- No error recovery (no acks). No retransmissions.
- Used by network management, DNS, Streamed multimedia (Applications that are loss tolerant, delay sensitive, or have their own reliability mechanisms)

Source Port	Dest Port	Length	Check-sum
16b	16b	16b	16b

← Size in bits

Error Detection: Checksum

- **Cyclic Redundancy Check (CRC)**: Powerful but generally requires hardware
- **Checksum**: Weak but easily done in software
  - **Example**: *1's complement* of 1’s complement sum of 16-bit words with overflow wrapped around



```
               1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
               1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
           ----------------------------------
wraparound   1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1
           ----------------------------------
       sum     1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0
  checksum     0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1
```

At receiver the sum is all 1’s and the checksum is zero.

1’s Complement

2’s Complement: -ve of a number is complement+1

- 1 = 0001   -1 = 1111
- 2 = 0010   -2 = 1110
- 0 = 0000   -0 = 0000

1’s complement: -ve of a number is it’s complement

- 1 = 0001   -1 = 1110
- 2 = 0010   -2 = 1101
- 0 = 0000   -0 = 1111

2’s Complement sum: Add with carry. Drop the final carry, if any.

6-7 = 0110 + (-0111) = 0110 + 1001 = 1111 => -1

1’s complement sum: Add with carry. Add end-around carry back to sum

- 6-7 = 0110 + (-0111) = 0110+1000 = 1110 => -1

Complement of 1’s complement sum: 0001

Checksum: At the transmitter: 0110 1000, append 0001

At the receiver: 0110 1000 0001 compute checksum of the full packet = complement of sum = complement of 1111 = 0000

Ref: https://en.wikipedia.org/wiki/Ones%27_complement

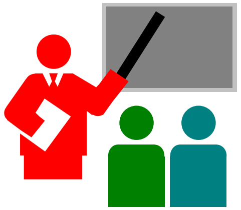
Homework 3A

Consider the following two 16-bit words: ABCD 1234

A. What is the checksum as computed by the sender

B. Add your answer of Part A to the end of the packet and show how the receiver will compute the checksum of the received three 16-bit words and confirm that there are no errors.

C. Now assume that the first bit of the packet is flipped due to an error. Repeat Part B at the receiver. Is the error detected?



UDP: Summary

1. UDP provides flow multiplexing using port #s
2. UDP optionally provides error detection using the checksum
3. UDP does not have error or loss recovery mechanism

Lab 3A: UDP

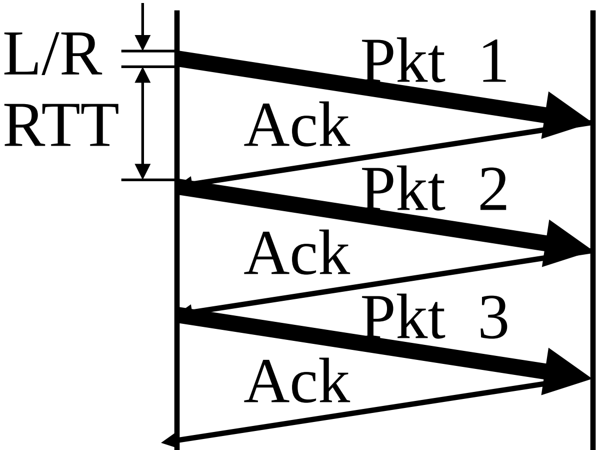
- Download the wireshark traces from http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip
- Open the trace file http-ethereal-trace-5 in Wireshark, View →Expand All and answer the following questions?

1. Examine the first packet. In the printout for this packet, select the line with UDP. The UPD header is highlighted in the trace. How many fields are there in the UDP header? Name these fields, their lengths and their values.
2. The value in the Length field is the length of what? Verify your claim with your captured UDP packet.
3. What is the maximum number of bytes that can be included in a UDP payload?
4. What is the largest possible source port number?
5. Examine the first two UDP packets. Describe the relationship between the port numbers in the two packets.

Flow Control

- Flow Control Goals:
  1. Sender does not flood the receiver,
  2. Maximize throughput

Stop and Wait Flow Control



L= Packet Length

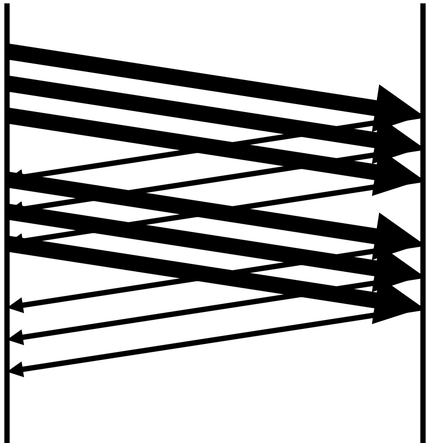
R= Link bit Rate

Large RTT

⇒ Low Thruput

$$\text{Throughput} = \frac{L/R}{RTT+L/R}$$

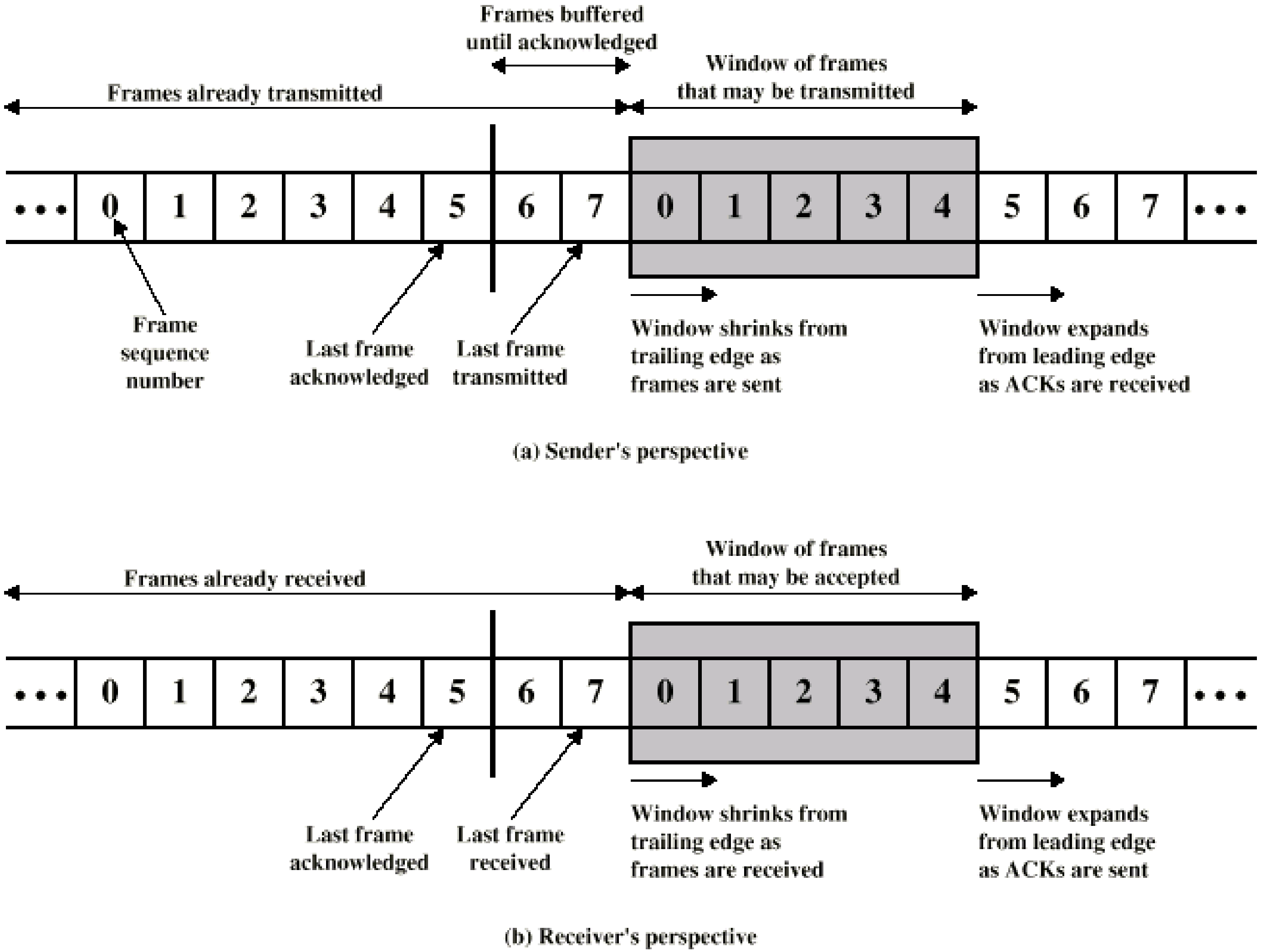
Window Flow Control



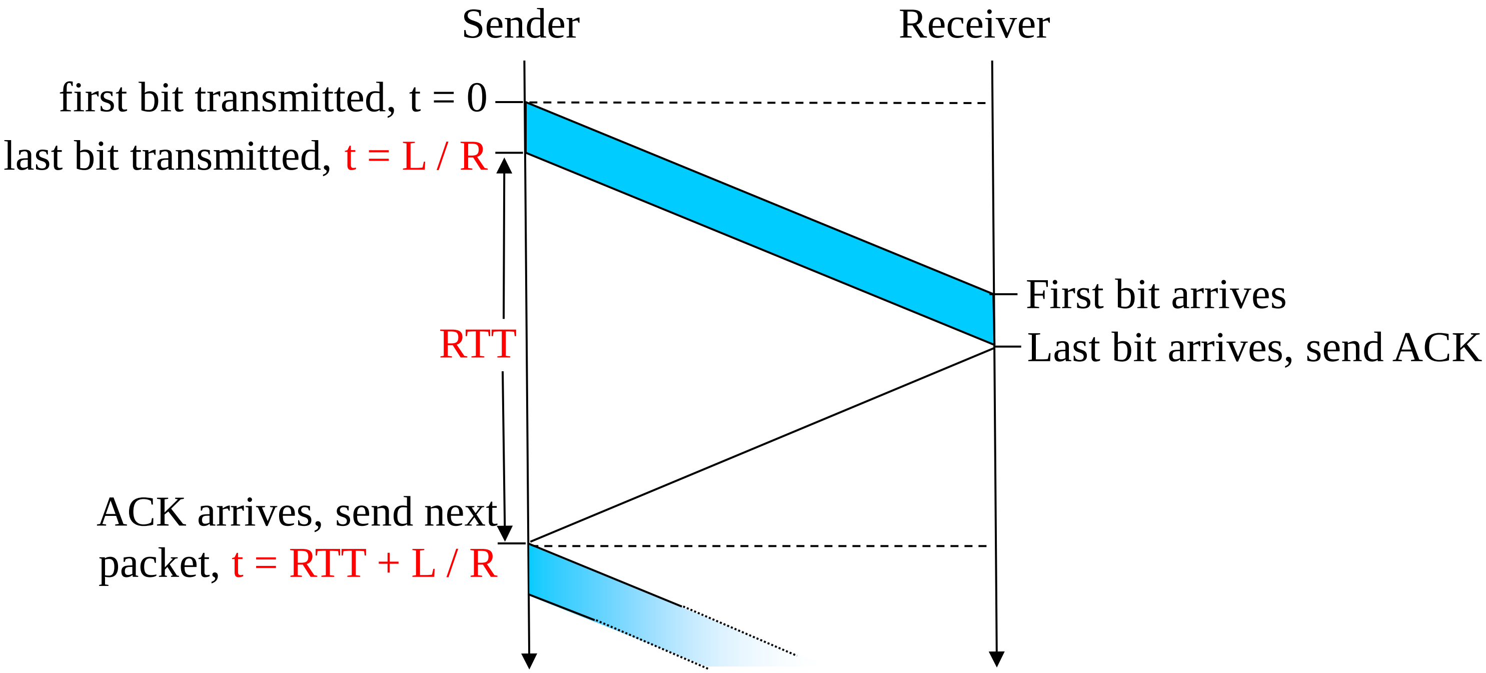
$$\text{Throughput} = \frac{W\ L/R}{RTT+L/R}$$

Ref: Textbook Section 3.4.2

Sliding Window Diagram


Frames buffered until acknowledged
Window of frames that may be transmitted
Frames already transmitted
• • • 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 • • •
Frame sequence number
Last frame acknowledged
Last frame transmitted
Window shrinks from trailing edge as frames are sent
Window expands from leading edge as ACKs are received
(a) Sender's perspective
Window of frames that may be accepted
Frames already received
• • • 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 • • •
Last frame acknowledged
Last frame received
Window shrinks from trailing edge as frames are received
Window expands from leading edge as ACKs are sent
(b) Receiver's perspective

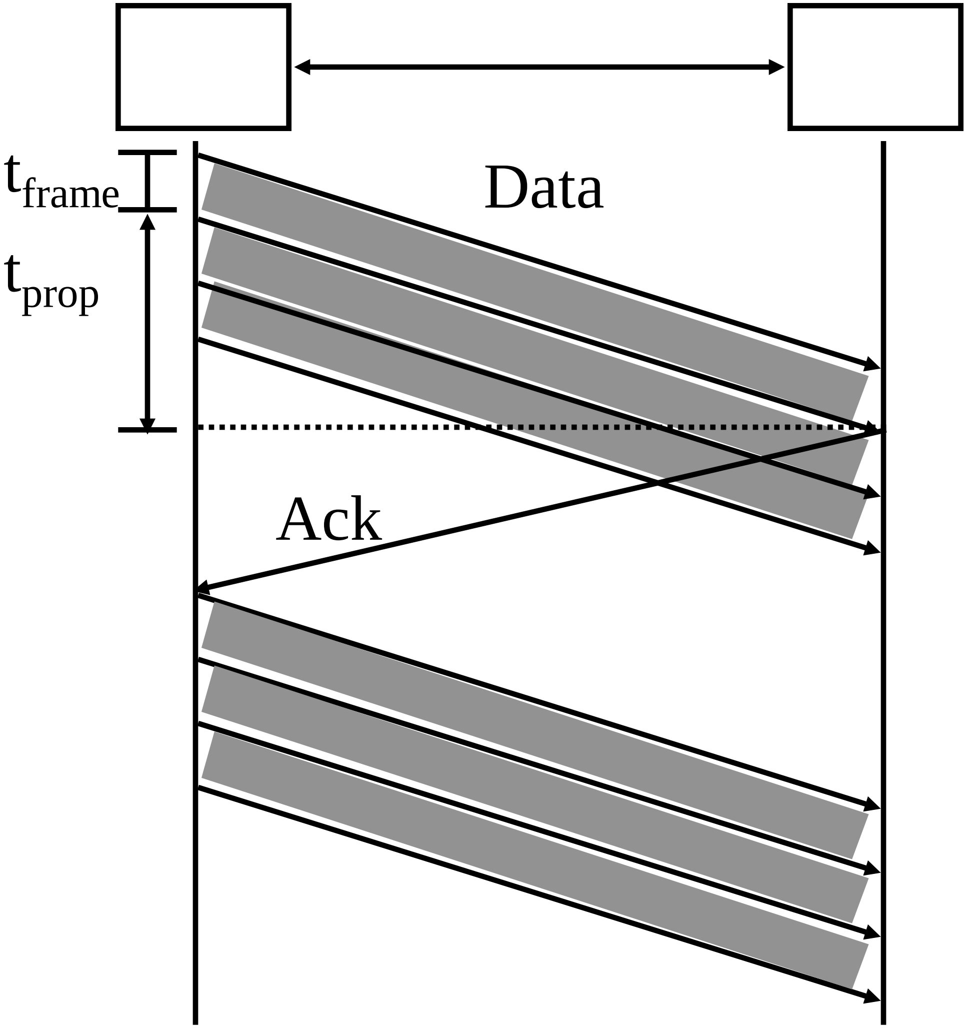
Stop and Wait Flow Control


Sender
Receiver
first bit transmitted, t = 0
last bit transmitted, t = L / R
RTT
First bit arrives
Last bit arrives, send ACK
ACK arrives, send next packet, t = RTT + L / R

$$U = \frac{L / R}{RTT + L / R} = \frac{t_{frame}}{2t_{prop}+t_{frame}} = \frac{1}{2\alpha + 1}$$

Here, $\alpha = t_{prop}/t_{frame}$

Sliding Window Protocol Efficiency



$$U = \frac{W\,t_{frame}}{2t_{prop} + t_{frame}}$$

$$= \begin{cases} \dfrac{W}{2\alpha+1} \\ 1 \text{ if } W > 2\alpha+1 \end{cases}$$

Here, $\alpha = t_{prop}/t_{frame}$

$W=1 \Rightarrow$ Stop and Wait

Utilization: Examples

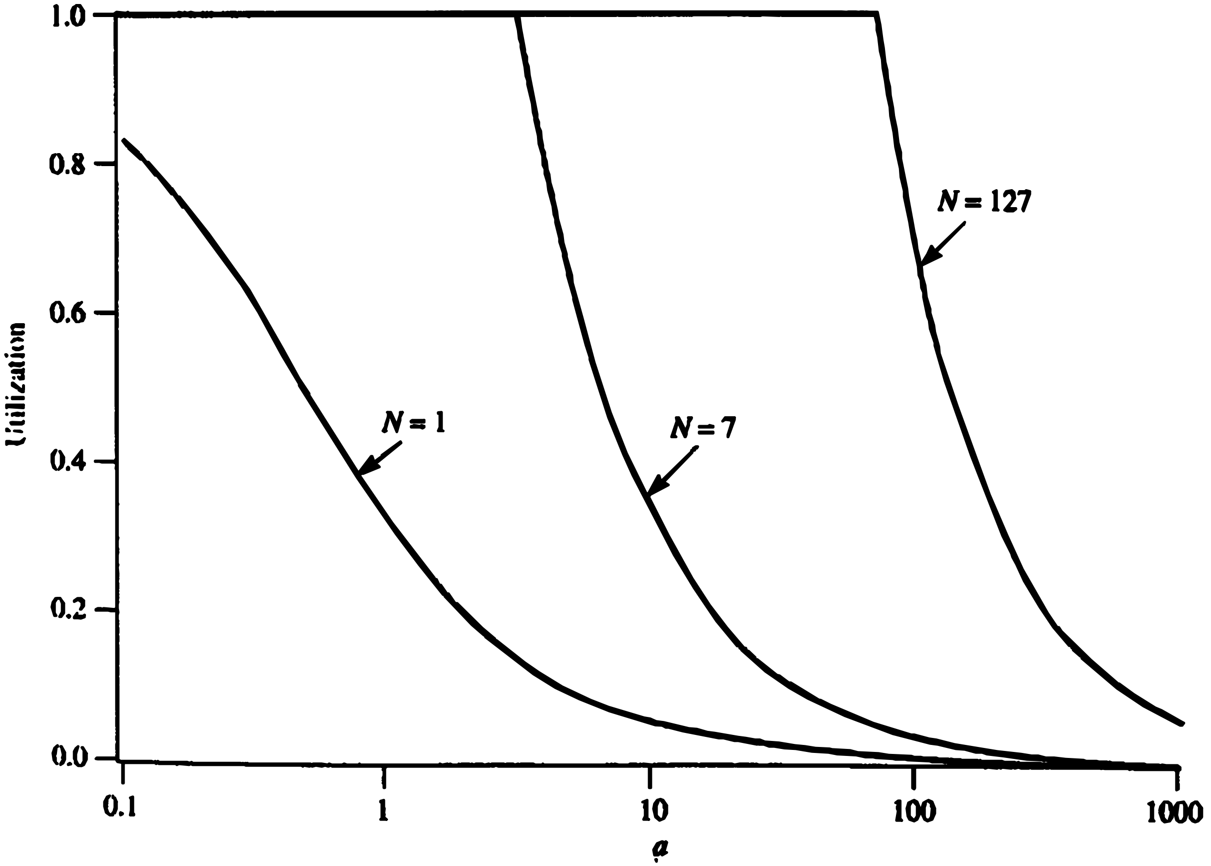
Satellite Link: One-way Propagation Delay = 270 ms
RTT=540 ms
Frame Size L = 500 Bytes = 4 kb
Data rate R = 56 kbps ⇒ t_{frame} = L/R= 4kb/56kbps =0.071s = 71 ms
$\alpha = t_{prop}/t_{frame} = 270/71 = 3.8$
$U = 1/(2\alpha+1) = 0.12$

- Short Link: 1 km = 5 μs (Assuming Fiber 200 m/μs),
  Rate=10 Mbps,
  Frame=500 bytes ⇒ t_{frame}= 4k/10M= 400 μs
  $\alpha=t_{prop}/t_{frame}=5/400=0.012 \Rightarrow U=1/(2\alpha+1)=0.98$

Note: The textbook uses RTT in place of t_{prop} and L/R for t_{frame}

Effect of Window Size

U

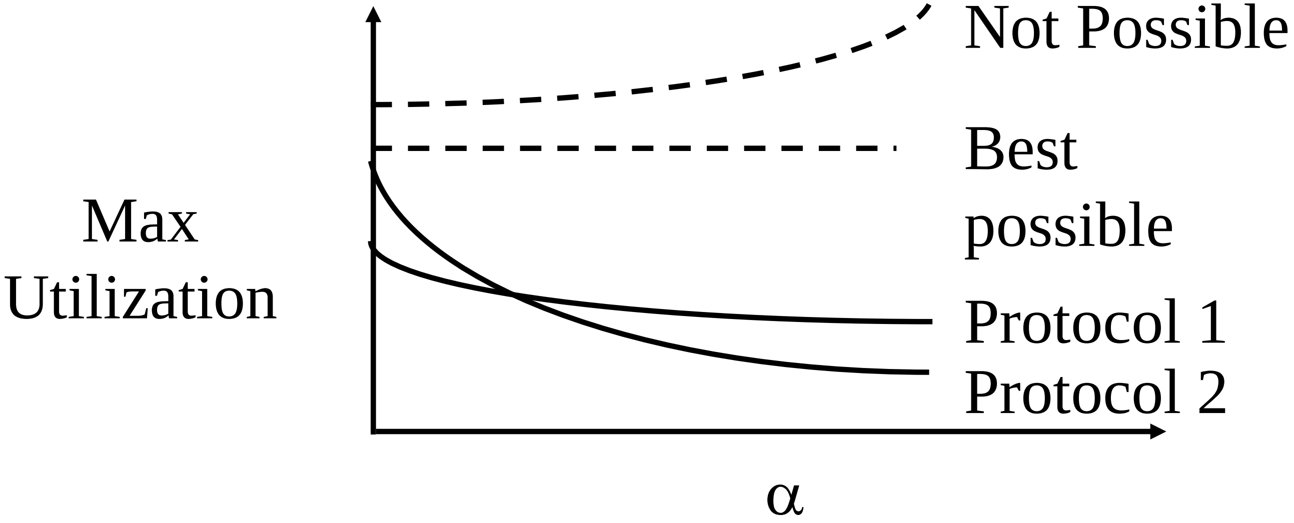


α

- Larger window is better for larger α

Efficiency Principle

- For **all** protocols, the maximum utilization (efficiency) is a *non-increasing* function of α.



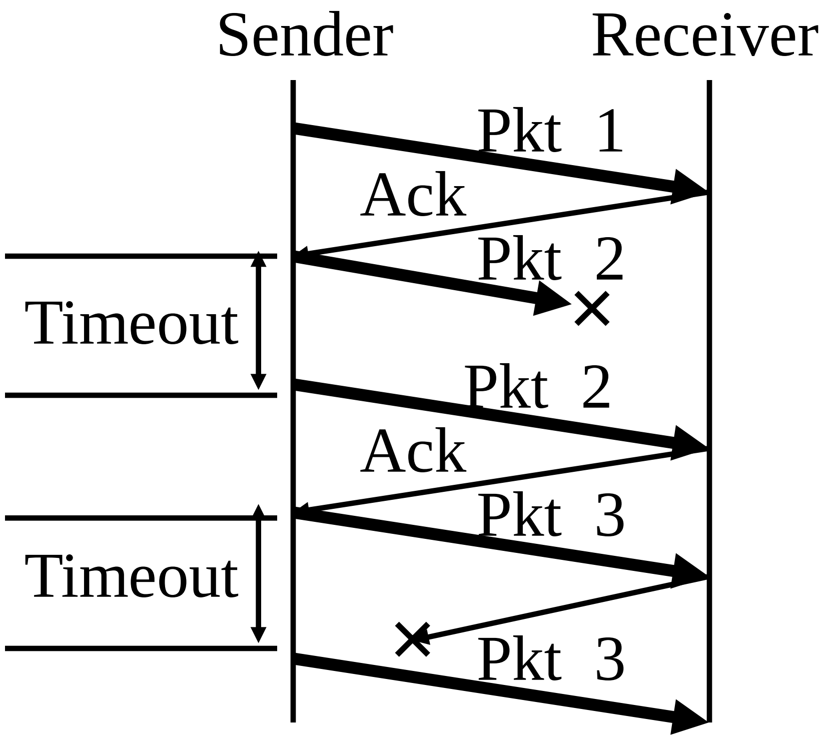
$$\alpha = \frac{t_{prop}}{t_{frame}} = \frac{\text{Distance/Speed of Signal}}{\text{Bits Transmitted /Bit rate}}$$

$$= \frac{\text{Distance} \times \text{Bit rate}}{\text{Bits Transmitted} \times \text{Speed of Signal}}$$

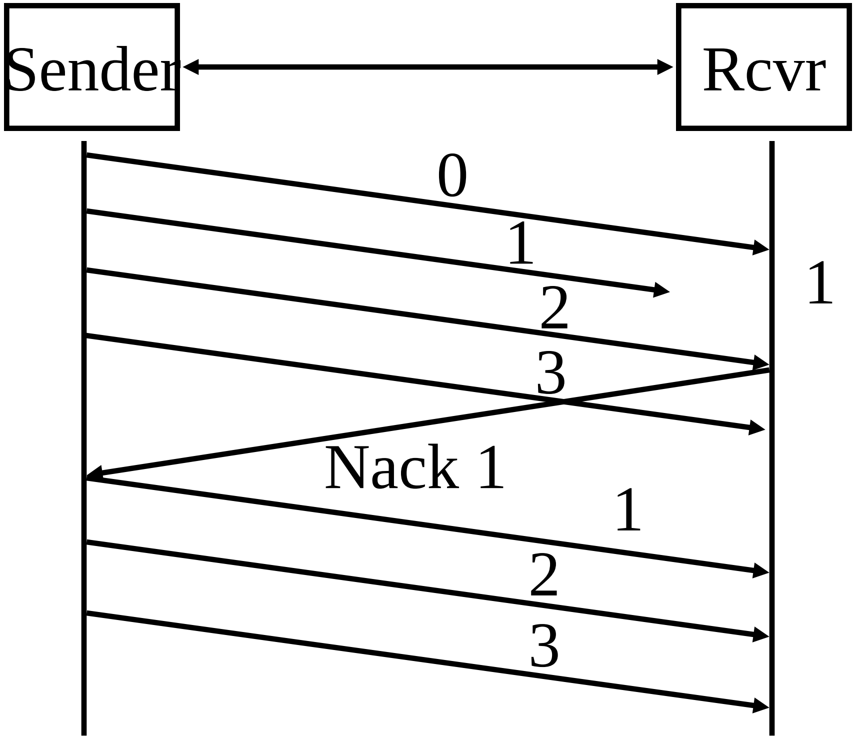
Error Control: Retransmissions

- Retransmit lost packets ⇒ **A**utomatic **R**epeat re**Q**uest (ARQ)

Stop and Wait ARQ

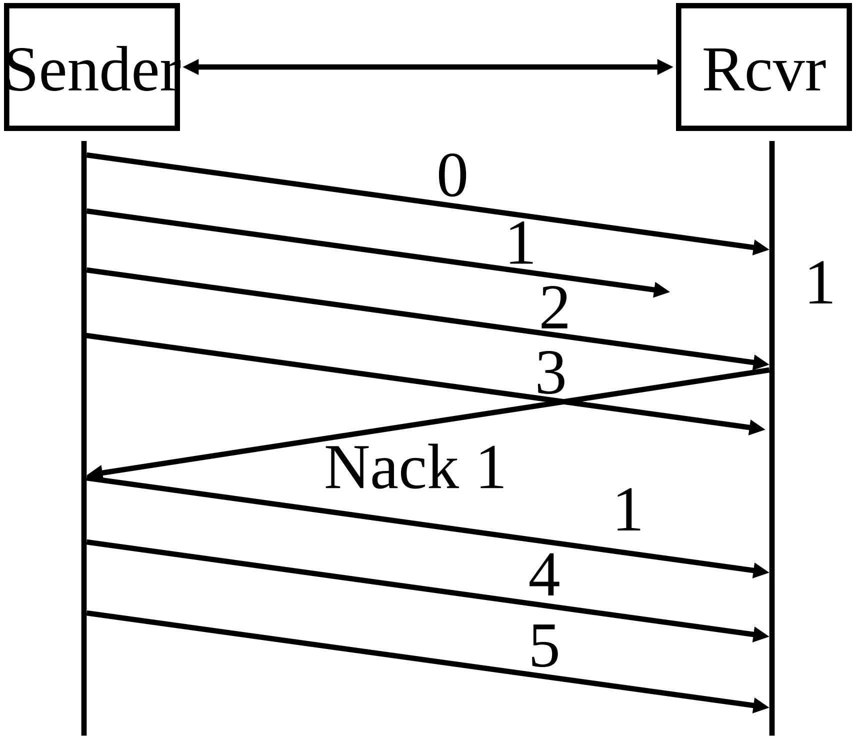


Go-Back-N ARQ



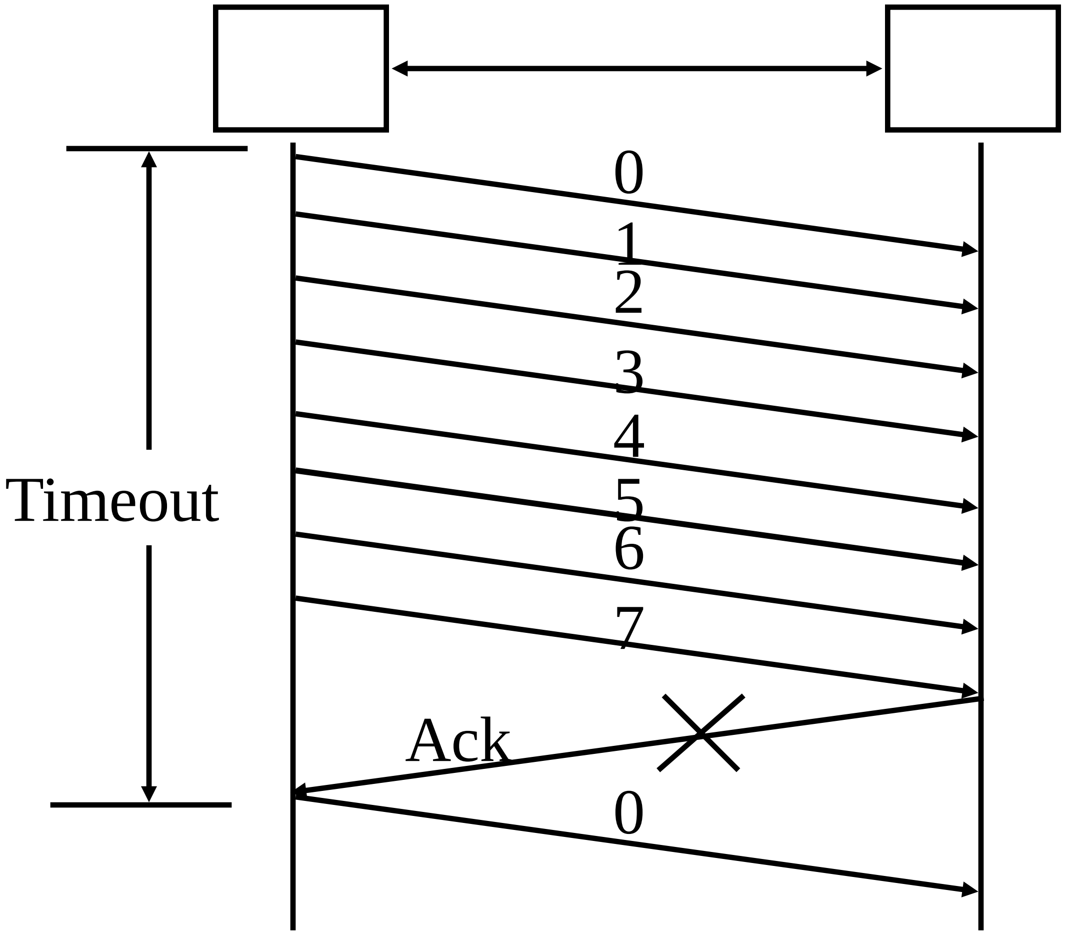
- Receiver does not cache out-of-order frames
- Sender has to ***go back*** and retransmit all frames after the lost frame

Selective Repeat ARQ



- Receiver caches out-of-order frames
- Sender retransmits only the lost frame
- Also known as selective *reject* ARQ

Selective Repeat: Window Size



Sequence number space $\geq$ 2 window size

Window size $\leq 2^{n-1}$

Performance: Maximum Utilization

- **Stop and Wait Flow Control**: $U = 1/(1+2\alpha)$
- **Window Flow Control**:

$$U = \begin{cases} 1 & W \geq 2\alpha+1 \\ W/(2\alpha+1) & W < 2\alpha+1 \end{cases}$$

- **Stop and Wait ARQ**: $U = (1-P)/(1+2\alpha)$
- **Go-back-N ARQ**:

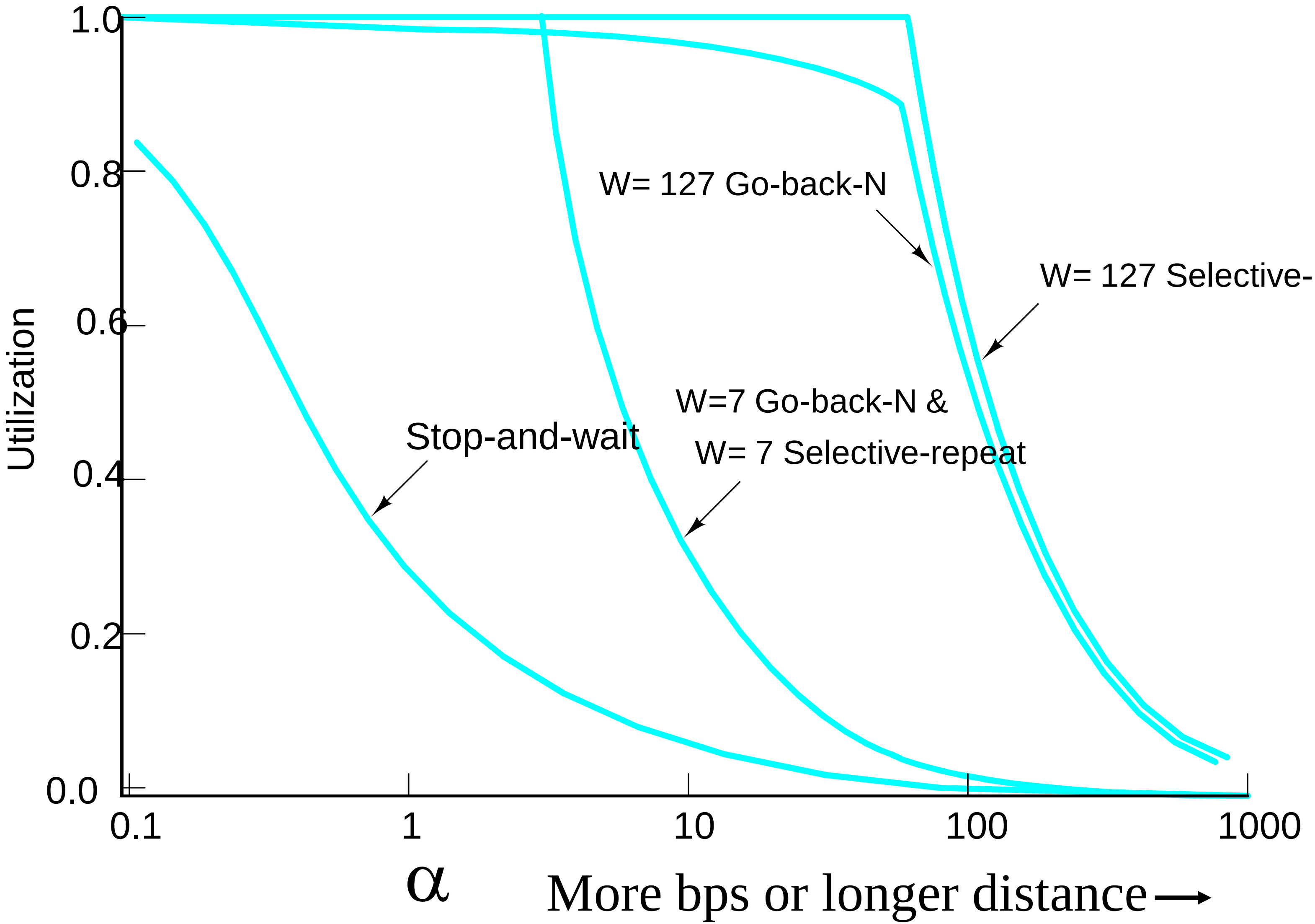
P = Probability of Loss

$$U = \begin{cases} (1-P)/(1+2\alpha P) & W \geq 2\alpha+1 \\ W(1-P)/[(2\alpha+1)(1-P+WP)] & W < 2\alpha+1 \end{cases}$$

- **Selective Repeat ARQ**:

$$U = \begin{cases} (1-P) & W \geq 2\alpha+1 \\ W(1-P)/(2\alpha+1) & W < 2\alpha+1 \end{cases}$$

Performance Comparison



Transport Layer Design Issues

1. Multiplexing/demultiplexing by a combination of source and destination IP addresses and port numbers.
2. Window flow control is better for long-distance or high-speed networks
3. Longer distance or higher speed
   ⇒ Larger α ⇒ Larger window is better
4. Stop and and wait flow control is ok for short distance or low-speed networks
5. Selective repeat is better than stop and wait ARQ
   Only slightly better than go-back-N

Homework 3B

Problem 19 on page 302 of the textbook:

Consider the GBN protocol with a sender window size of 3 and a sequence number range of 1,024. Suppose that at time t, the next in-order packet that the receiver is expecting has a sequence number of k. Assume that the medium does not reorder messages. Answer the following questions:

A. What are the possible sets of sequence numbers insdie the sender's window at time t? Justify your answer.

B. What are all possible values of the ACK field in all possible messages currently propagating back to the sender at time t? Justify your answer.

Window Flow Control:

C. How big window (in number of packets) is required for the channel utilization to be greater than 60% on a cross-country link of 4000 km running at 20 Mbps using 1 kByte packets?

Efficiency Principle:

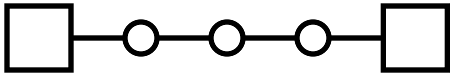
D. Ethernet V1 access protocol was designed to run at 10 Mbps over 2.5 Km using 1500 byte packets. This same protocol needs to be used at 100 Mbps at the same efficiency. What distance can it cover if the frame size is not changed?

TCP

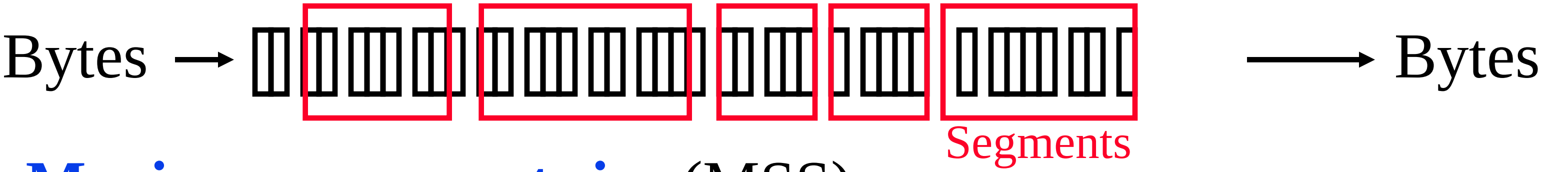
1. TCP Header Format, Options, Checksum
2. TCP Connection Management
3. Round Trip Time Estimation
4. Principles of Congestion Control
5. Slow Start Congestion Control

Key Features of TCP

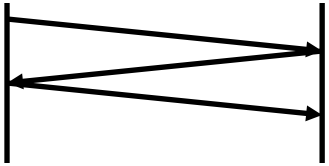
- **Point-to-Point**: One sender, one receiver



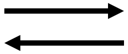
- **Byte Stream**: No message boundaries.
  TCP makes "segments"



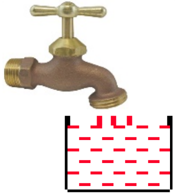
- **Maximum segment size** (MSS)
- **Connection Oriented**: Handshake to initialize states before data exchange



- **Full Duplex**: Bidirectional data flow in one connection



- **Reliable**: In-order byte delivery

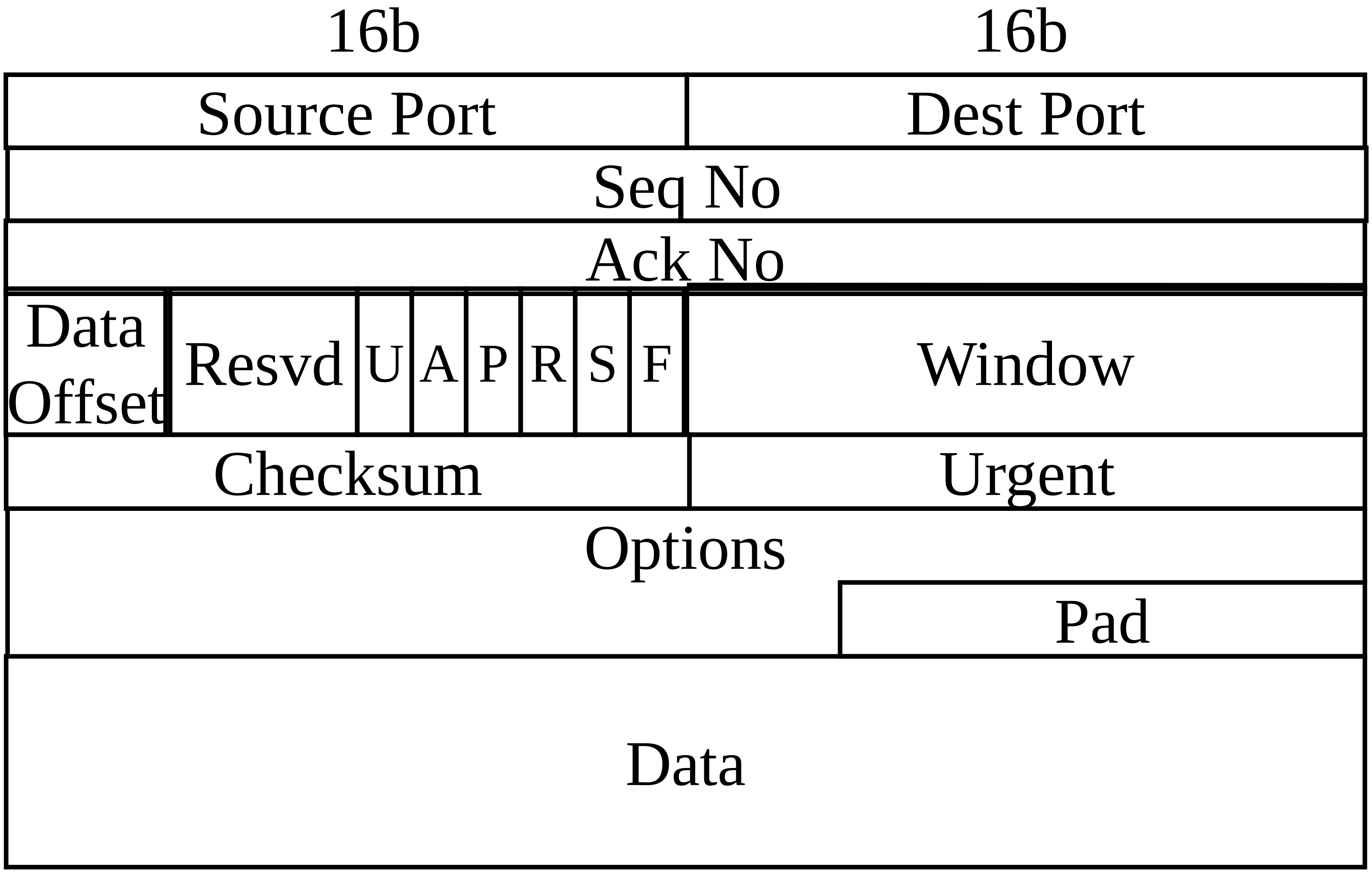


- **Flow control**: To avoid receiver buffer overflow
- **Congestion control**: To avoid network router buffer overflow

TCP

- Transmission Control Protocol
- Key Services:
  - **Send**: Please send when convenient
  - **Data stream push**: Destination TCP, please deliver it immediately to the receiving application.
    ⇒ Source TCP, please send it now.
    Set on last packet of an application message.
  - **Urgent data signaling**: Destination TCP, please give this urgent data to the user out-of-band.
    Generally used for CTRL-C.

TCP Segment Format (Cont)



16b

16b

Source Port

Dest Port

Seq No

Ack No

Data Offset

Resvd

U

A

P

R

S

F

Window

Checksum

Urgent

Options

Pad

Data

TCP Header Fields

- **Source Port** (16 bits): Identifies source user process
- **Destination Port** (16 bits)
  21 = FTP, 23 = Telnet, 53 = DNS, 80 = HTTP, ...
- **Sequence Number** (32 bits): Sequence number of the first byte in the segment. If SYN is present, this is the initial sequence number (ISN) and the first data byte is ISN+1.
- **Ack number** (32 bits): Next byte expected
- **Data offset** (4 bits): Number of 32-bit words in the header
- **Reserved** (6 bits)

TCP Header (Cont)

- **Control** (6 bits):Urgent pointer field significant,
  Ack field significant,
  Push function,
  Reset the connection,
  Synchronize the sequence numbers,
  No more data from sender

URG	ACK	PSH	RST	SYN	FIN

- **Window** (16 bits):
  Will accept [Ack] to [Ack]+[window]-1

TCP Header (Cont)

- **Checksum** (16 bits): covers the segment plus a pseudo header. Includes the following fields from IP header: source and dest adr, protocol, segment length. Protects from IP misdelivery.
- **Urgent pointer** (16 bits): Points to the byte following urgent data. Lets receiver know how much data it should deliver right away out-of-band.
- **Options** (variable):
  Max segment size (does not include TCP header, default 536 bytes), Window scale factor, Selective Ack permitted, Timestamp, No-Op, End-of-options

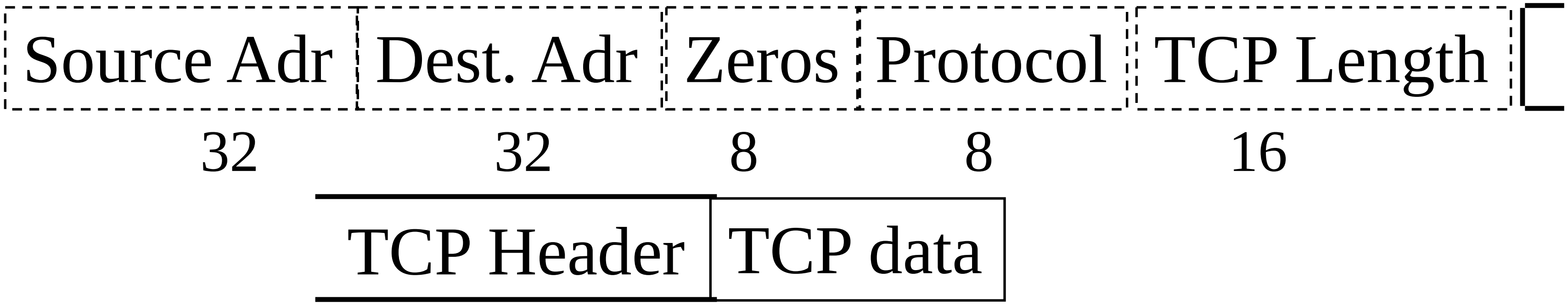
TCP Options

Kind	Length	Meaning
0	1	End of Valid options in header
1	1	No-op
2	4	Maximum Segment Size
3	3	Window Scale Factor
8	10	Timestamp

- **End of Options**: Stop looking for further option
- **No-op**: Ignore this byte. Used to align the next option on a 4-byte word boundary
- **Max Segment Size (MSS):** Does not include TCP header

TCP Checksum

- Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the TCP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.
- Checksum field is filled with zeros initially
- TCP length (in octet) is not transmitted but used in calculations.
- Efficient implementation in RFC1071.

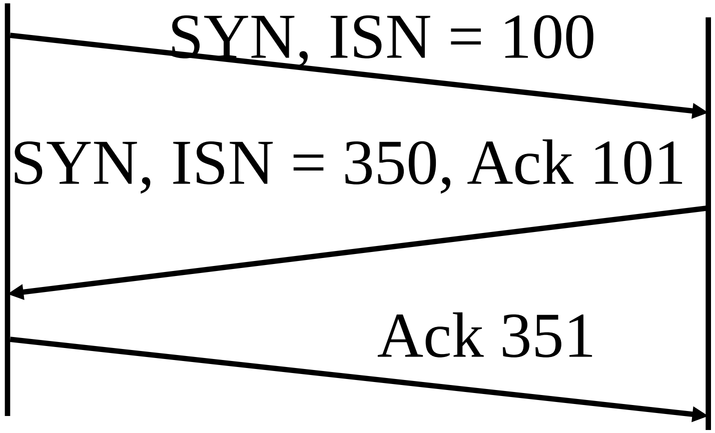


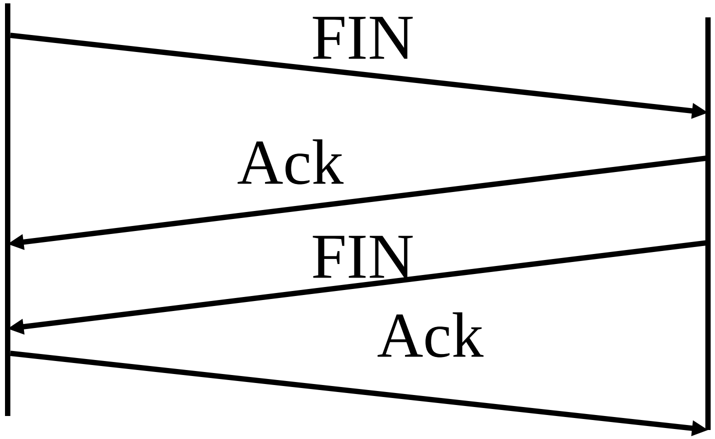
Source Adr	Dest. Adr	Zeros	Protocol	TCP Length
32	32	8	8	16

TCP Header	TCP data

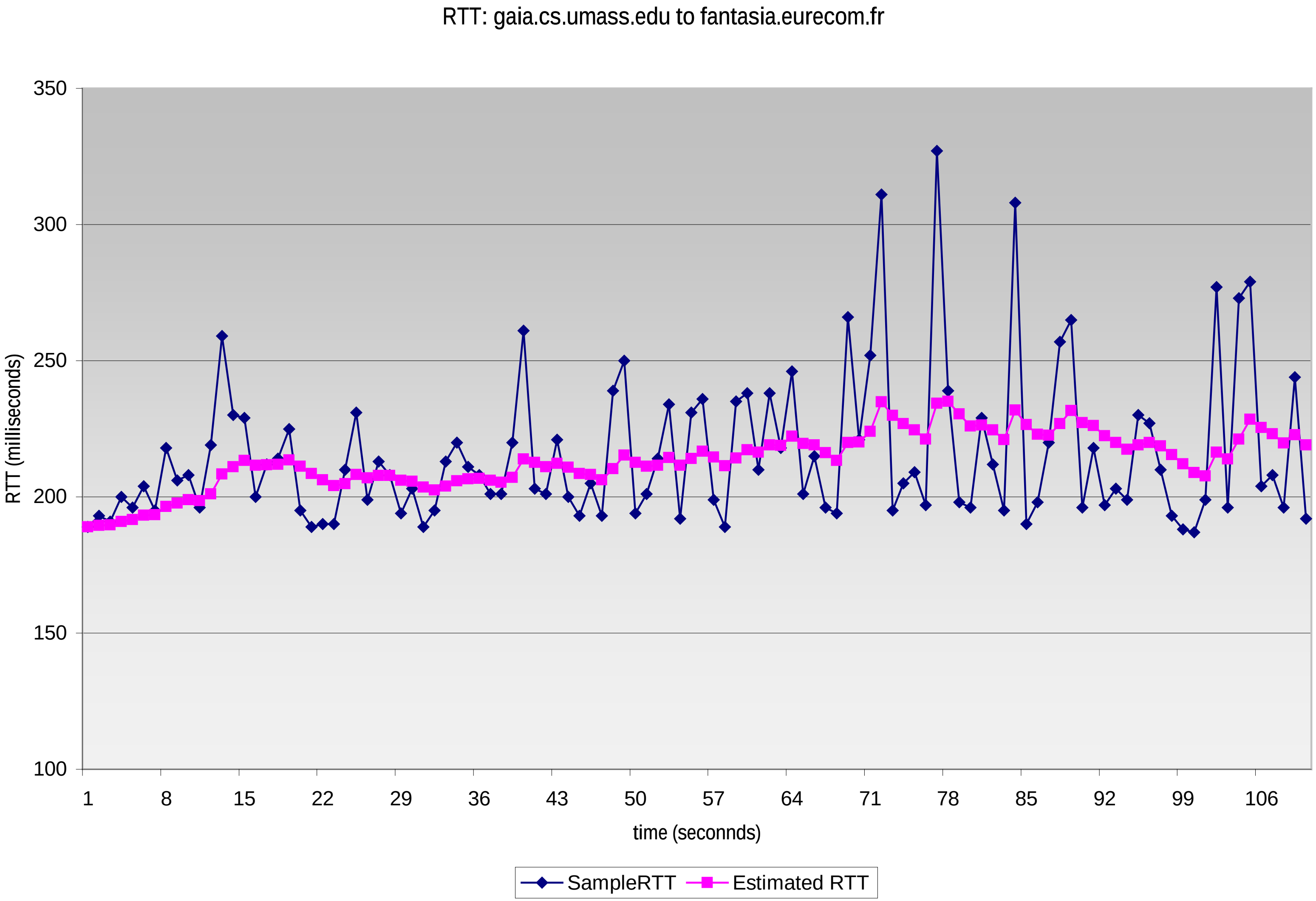
TCP Connection Management

- Connection Establishment
  - Three way handshake
  - SYN flag set
    ⇒ Request for connection
- Connection Termination
  - Close with FIN flag set
  - Abort



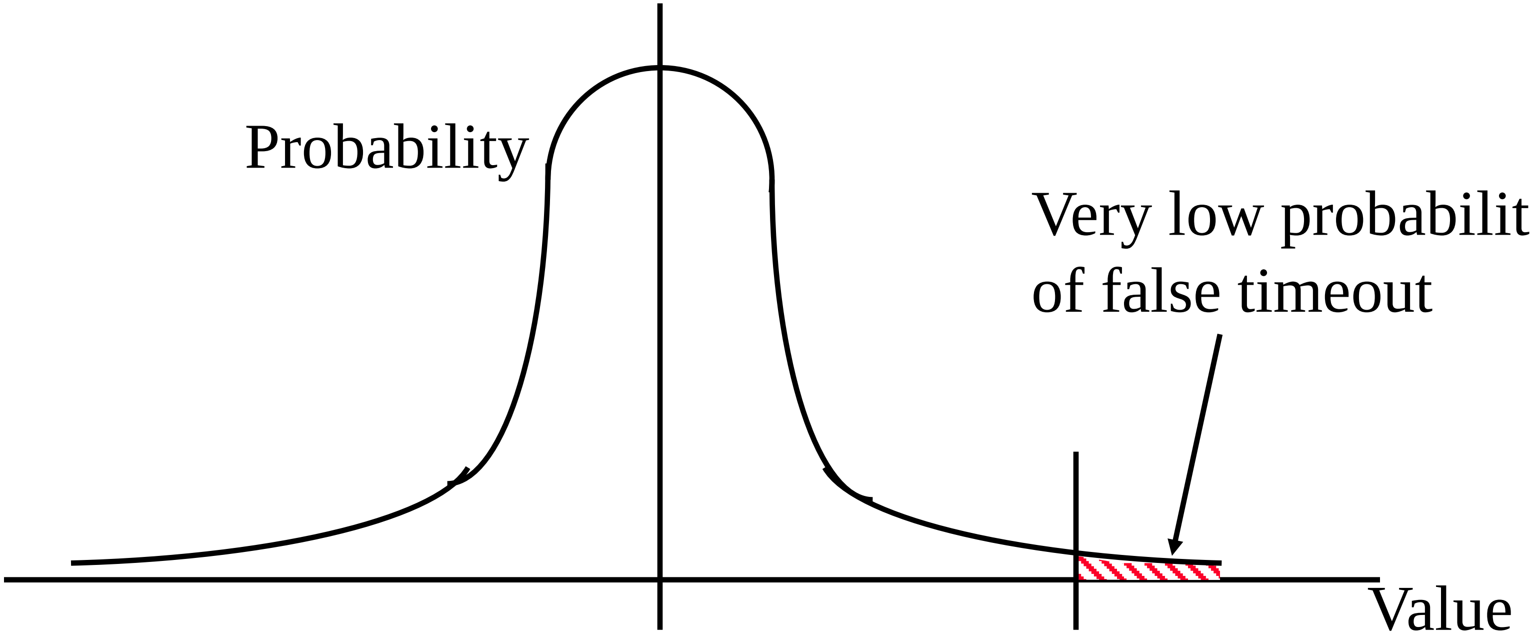


Example RTT estimation:

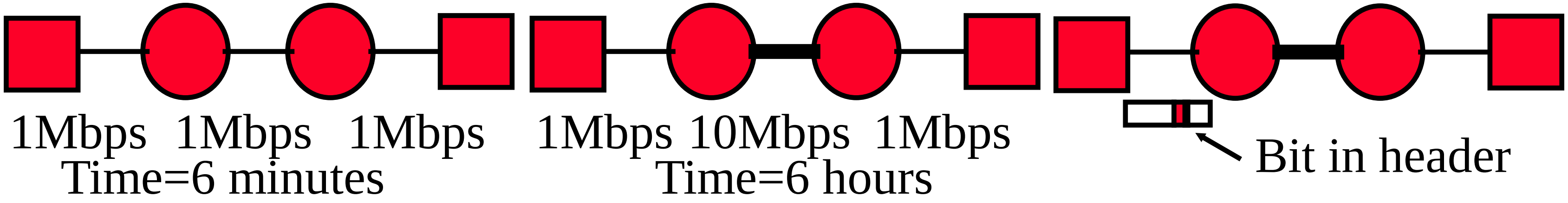


Round Trip Time Estimation

- Measured round trip time (SampleRTT) is very random.
- EstimatedRTT=(1- α)EstimatedRTT+α SampleRTT
- DevRTT = (1-β)DevRTT+ β |SampleRTT-EstmatedRTT|
- TimeoutInterval=EstimatedRTT+**4** DevRTT



Our Research on Congestion Control



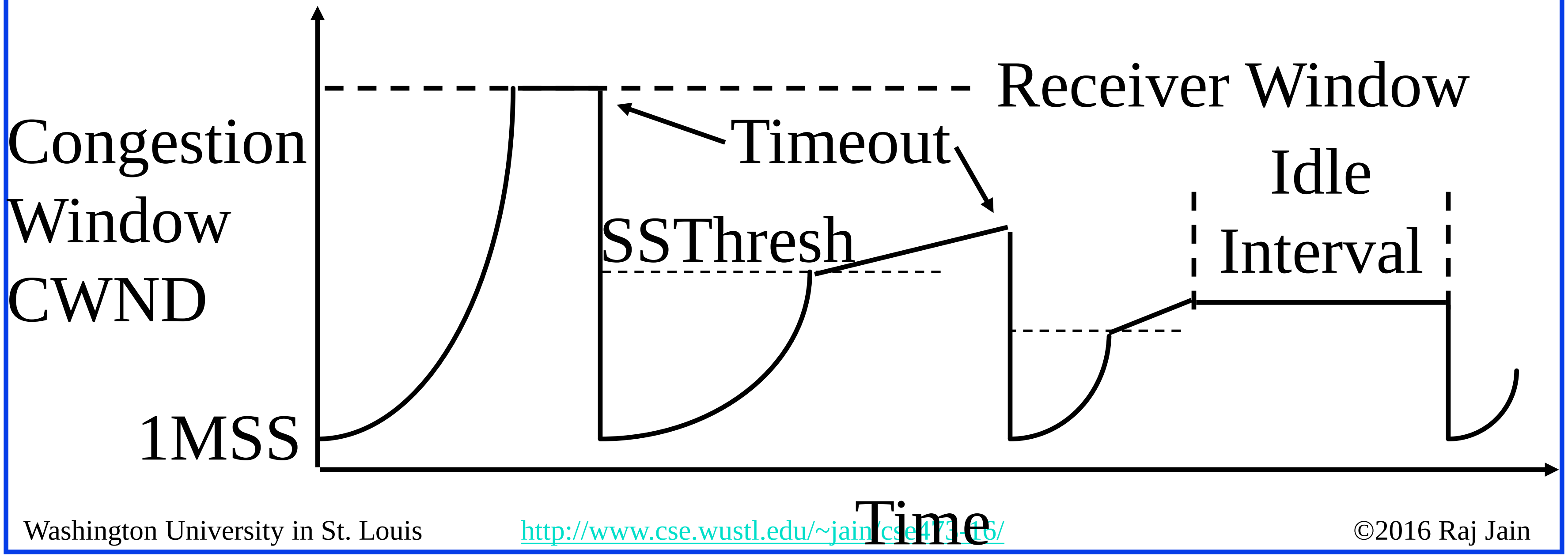
- ❑ Early 1980s Digital Equipment Corporation (DEC) introduced Ethernet products
- ❑ Noticed that throughput goes down with a higher-speed link in middle (because no congestion mechanisms in TCP)
- ❑ Results:
  1. Timeout ⇒ Congestion
     ⇒ Reduce the TCP window to one on a timeout [Jain 1986]
  2. Routers should set a bit when congested (DECbit). [Jain, Ramakrishnan, Chiu 1988]
  3. Introduced the term "Congestion Avoidance"
  4. Additive increase and multiplicative decrease (AIMD principle) [Chiu and Jain 1989]
- ❑ There were presented to IETF in 1986.
  ⇒ Slow-start based on Timeout and AIMD [Van Jacobson 1988]

Slow Start Congestion Control

- Window = Flow control avoids receiver overrun
- Need congestion control to avoid network overrun
- The sender maintains two windows:
  Credits from the receiver
  Congestion window from the network
  Congestion window is always less than the receiver window
- Starts with a congestion window (CWND) of 1 max segment size (MSS)
  ⇒ Do not disturb existing connections too much.
- Increase CWND by 1 MSS every time an ack is received
- Assume CWND is in bytes

Slow Start (Cont)

- If segments lost, remember slow start threshold (SSThresh) to CWND/2
  Set CWND to 1 MSS
  Increment by 1MSS per ack until SSThresh
  Increment by 1 MSS*MSS/CWND per ack afterwards

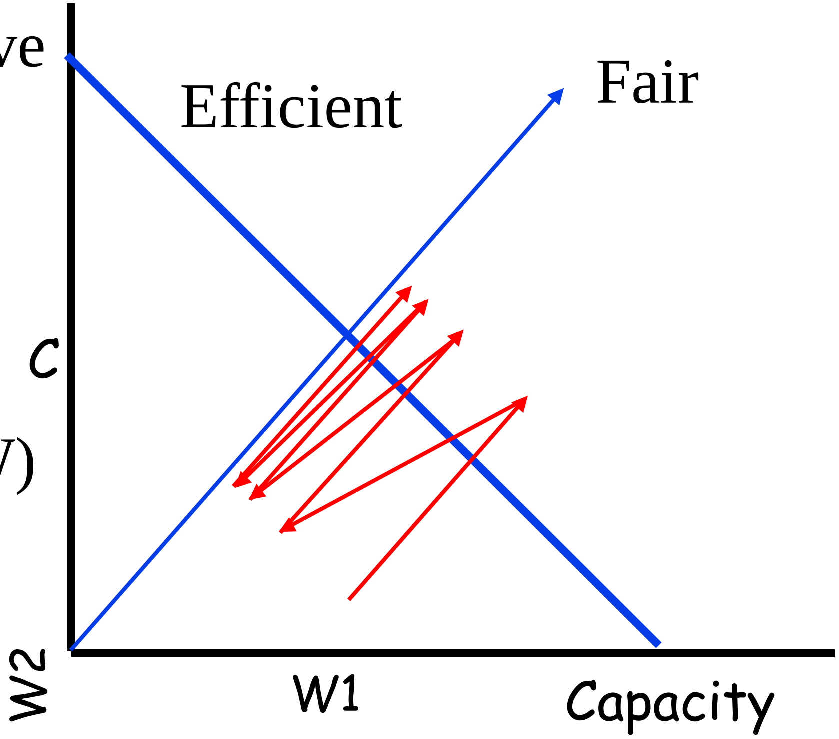


Slow Start (Cont)

- At the beginning, SSThresh = Receiver window
- After a long idle period (exceeding one round-trip time), reset the congestion window to one.
- If CWND is W MSS, W acks are received in one round trip.
- Below SSThresh, CWND is increased by 1MSS on every ack
  ⇒ CWND increases to 2W MSS in one round trip
  ⇒ CWND increases exponentially with time
  Exponential growth phase is also known as "***Slow start***" phase
- Above SSThresh, CWND is increased by MSS/CWND on every ack
  ⇒ CWND increases by 1 MSS in one round trip
  ⇒ CWND increases linearly with time
  The linear growth phase is known as "***congestion avoidance"*** phase

AIMD Principle

- Additive Increase, Multiplicative Decrease
- W1+W2 = Capacity ⇒ Efficiency, W1=W2 ⇒ Fairness
- (W1,W2) to (W1+ΔW,W2+ΔW) ⇒ Linear increase (45° line)
- (W1,W2) to (kW1,kW2) ⇒ Multiplicative decrease (line through origin)

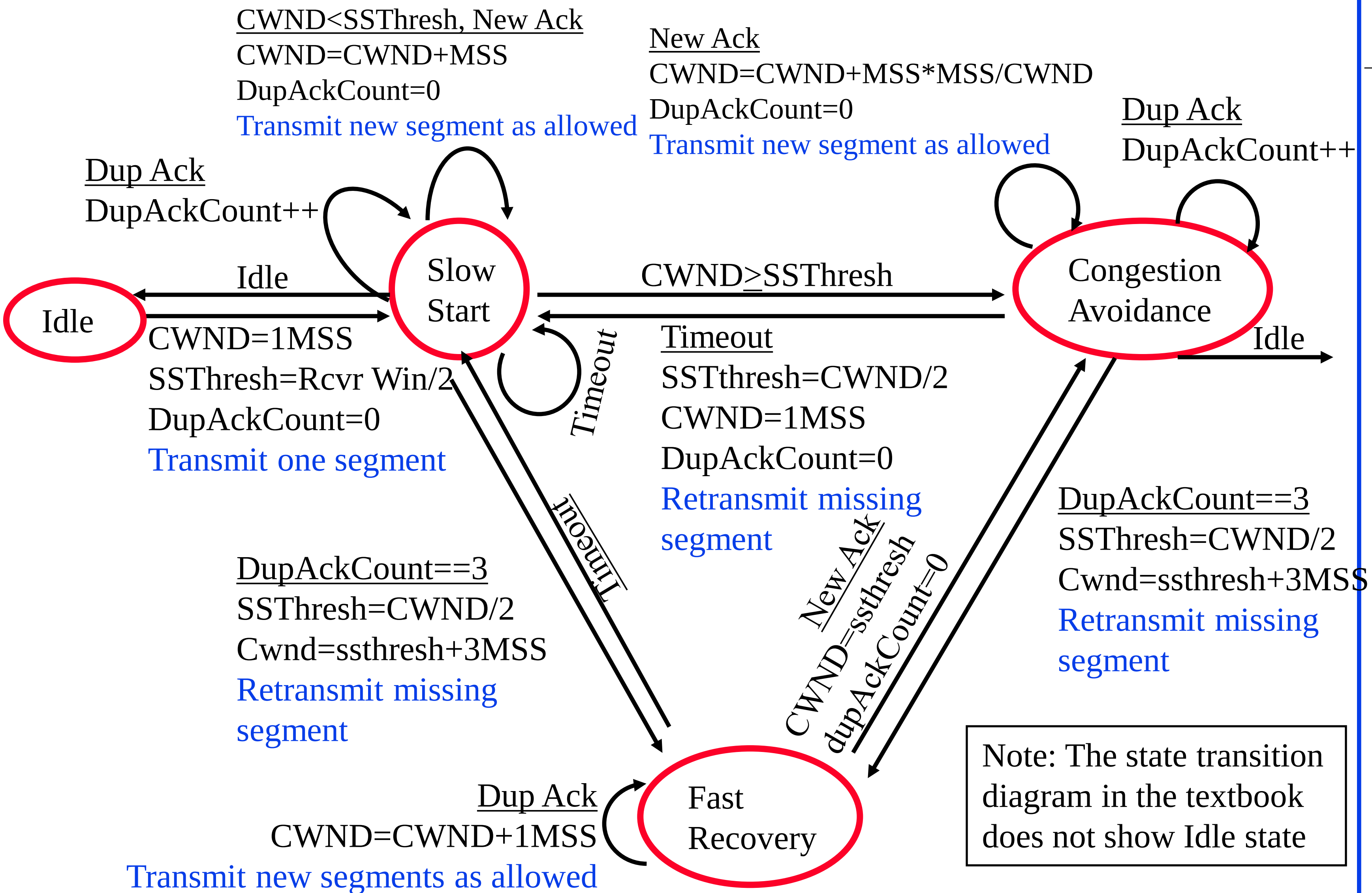


Ref: D. Chiu and Raj Jain, "**Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks**," Journal of Computer Networks and ISDN, Vol. 17, No. 1, June 1989, pp. 1-14, http://www.cse.wustl.edu/~jain/papers/cong_av.htm

Fast Retransmit

- Optional – implemented in TCP Reno
  (Earlier version was TCP Tahoe)
- Duplicate Ack indicates a lost/out-of-order segment
- On receiving 3 duplicate acks (4th ack for the same segment):
  - Enter Fast Recovery mode
    - Retransmit missing segment
    - Set SSThresh=CWND/2
    - Set CWND=SSThresh+3 MSS
    - Every subsequent duplicate ack: CWND=CWND+1MSS
  - When a new ack (not a duplicate ack) is received
    - Exit fast recovery
    - Set CWND=SSTHRESH

TCP Congestion Control State Diagram



CWND<SSThresh, New Ack
CWND=CWND+MSS
DupAckCount=0
Transmit new segment as allowed

New Ack
CWND=CWND+MSS*MSS/CWND
DupAckCount=0
Transmit new segment as allowed

Dup Ack
DupAckCount++

Dup Ack
DupAckCount++

Idle

Slow Start

CWND≥SSThresh

Congestion Avoidance

Idle

Idle

CWND=1MSS
SSThresh=Rcvr Win/2
DupAckCount=0
Transmit one segment

Timeout

Timeout
SSTthresh=CWND/2
CWND=1MSS
DupAckCount=0
Retransmit missing segment

Timeout

New Ack
CWND=ssthresh
dupAckCount=0

DupAckCount==3
SSThresh=CWND/2
Cwnd=ssthresh+3MSS
Retransmit missing segment

DupAckCount==3
SSThresh=CWND/2
Cwnd=ssthresh+3MSS
Retransmit missing segment

Dup Ack
CWND=CWND+1MSS
Transmit new segments as allowed

Fast Recovery

Note: The state transition diagram in the textbook does not show Idle state

TCP Average Throughput

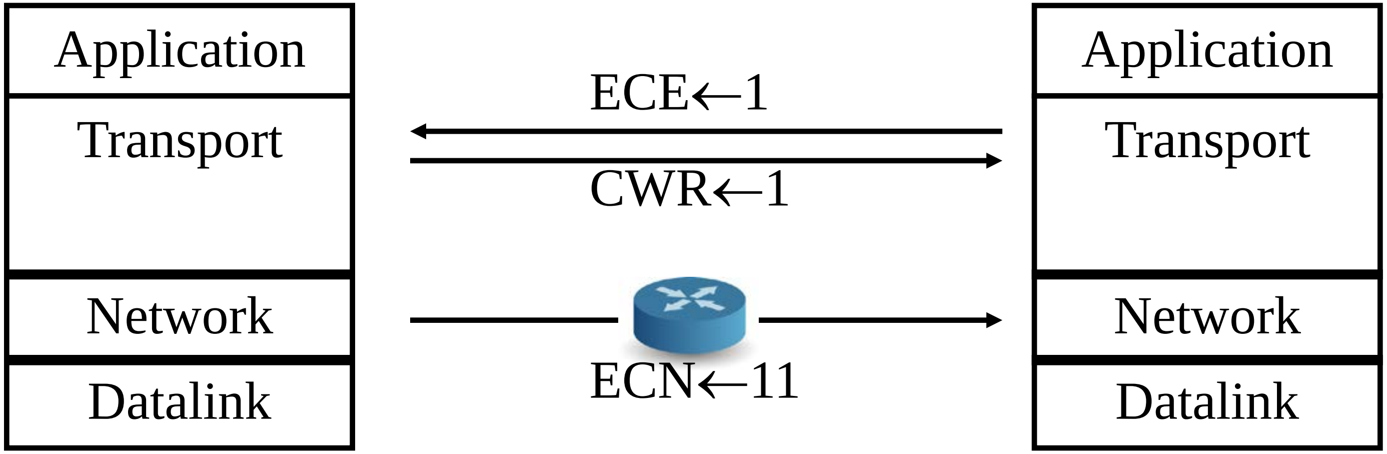
- Average Throughput = $\dfrac{1.22\ \text{MSS}}{\text{RTT}\ \sqrt{P}}$

- Here, P = Probability of Packet loss

- Note 1: The formula is an approximation which does not apply at P=0 or P=1. At P=1, the throughput is zero. At P=0, the throughput is min{1, (Receiver Window/RTT)}
- Note 1: The textbook uses L for probability of packet loss but it was used earlier for length of packets.

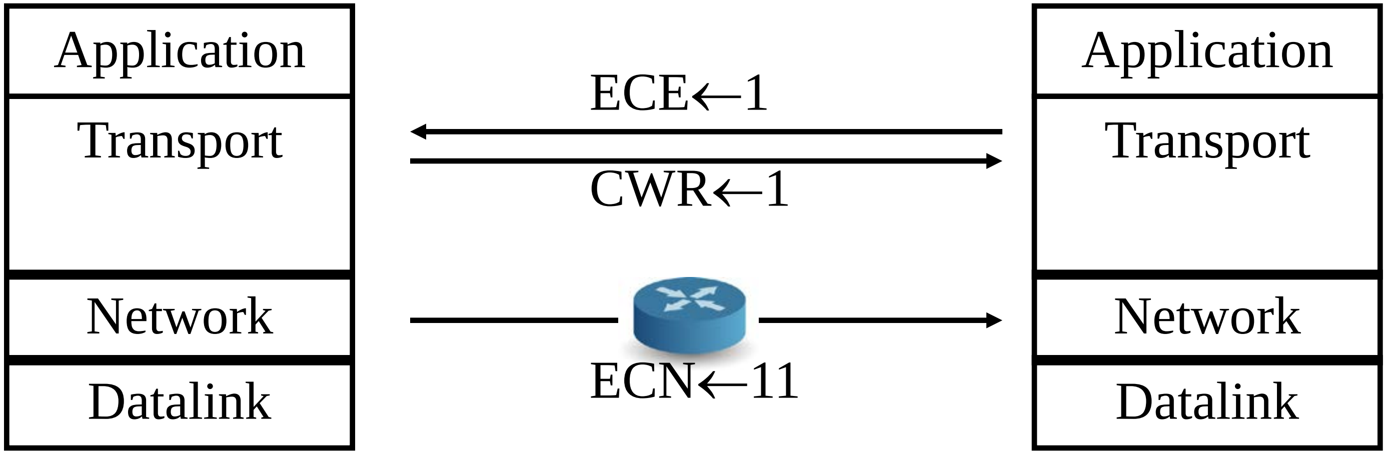
Explicit Congestion Notification (ECN)

- Explicit congestion notification (ECN) is based on our DECbit research. Two bits in IP Header:
  - 00: Transport is not capable of ECN (e.g., UDP)
  - 01: Transport is capable of ECN
  - 10: Transport is capable of ECN
  - 11: Congestion experienced (CE)
- When a router encounters congestion, instead of dropping the datagram, it marks the two bits as "11" congestion experienced



ECN (Cont)

- On receiving “CE” code point, the receiver sends “ECN Echo (ECE)” flag in the TCP header
- On seeing the ECE flag, the source reduces its congestion window, and sets “Congestion Window Reduced (CWR) flag in outgoing segment
- On receiving “CWR” flag, the receiver, stops setting ECE bit



Ref: https://en.wikipedia.org/wiki/Explicit_Congestion_Notification

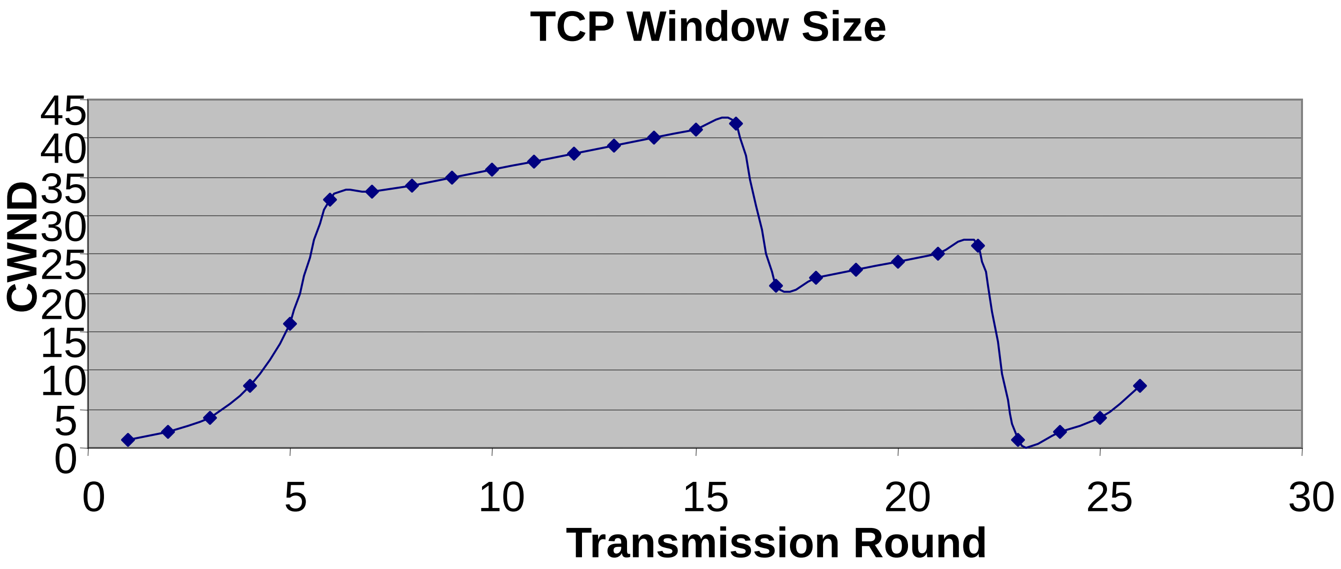
TCP: Summary

1. TCP uses port numbers for multiplexing
2. TCP provides reliable full-duplex connections.
3. TCP is stream based and has window flow control
4. Slow-start congestion control works on timeout
5. Explicit congestion notification works using ECN bits

Homework 3C

- Consider Figure below. Assuming TCP Reno is the protocol experiencing the behavior shown above, answer the following questions. In all cases, you should provide a short discussion justifying your answer.

Round	CWND
1	1
2	2
3	4
4	8
5	16
6	32
7	33
8	34
9	35
10	36
11	37
12	38
13	39
14	40
15	41
16	42
17	21
18	22
19	23
20	24
21	25
22	26
23	1
24	2
25	4
26	8



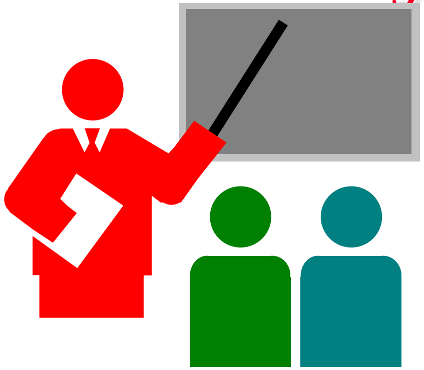
Homework 3C (Cont)

- A. Identify the interval of time when TCP slow start is operating.
- B. Identify the intervals of time when TCP congestion avoidance is operating.
- C. After the 16th transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?
- D. After the 22nd transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?
- E. What is the initial value of ssthresh at the first transmission round?
- F .What is the value of ssthresh at the 18th transmission round?
- G. What is the value of ssthresh at the 24th transmission round?

Homework 3C (Cont)

- H. During what transmission round is the 70th segment sent?
- I. Assuming a packet loss is detected after the 26th round by the receipt of a triple duplicate ACK, what will be the values of the congestion window size and of ssthresh?
- J. Suppose TCP Tahoe is used (instead of TCP Reno), and assume that triple duplicate ACKs are received at the 16th round. What are the ssthresh and the congestion window size at the 19th round?
- K. Again suppose TCP Tahoe is used, and there is a timeout event at the end of 22nd round. How many packets have been sent out from 17th round till 22nd round, inclusive?

Summary



1. Multiplexing/demultiplexing by a combination of source and destination IP addresses and port numbers.
2. Longer distance or higher speed
   ⇒ Larger α ⇒ Larger window is better
3. Window flow control is better for long-distance or high-speed networks
4. UDP is connectionless and simple.
   No flow/error control. Has error detection.
5. TCP provides full-duplex connections with flow/error/congestion control.

Lab 3B: TCP

- In Wireshark, open the trace file tcp-ethereal-trace-1 in http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip and answer the following:

1. What is the IP address and TCP port number used by the client computer (source) to transfer the file to gaia.cs.umass.edu?
2. What is the IP address and port number used by gaia.cs.umass.edu to receive the file.
3. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?
4. What is the sequence number of the SYN Ack segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the ACKnowledgement field in the SYN Ack segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYN Ack segment?

Lab 3B: TCP (Cont)

5. What is the sequence number of the TCP segment containing the HTTP POST command? Note that in order to find the POST command, you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.
6. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received? Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the RTT value for each of the six segments?

Lab 3B: TCP (Cont)

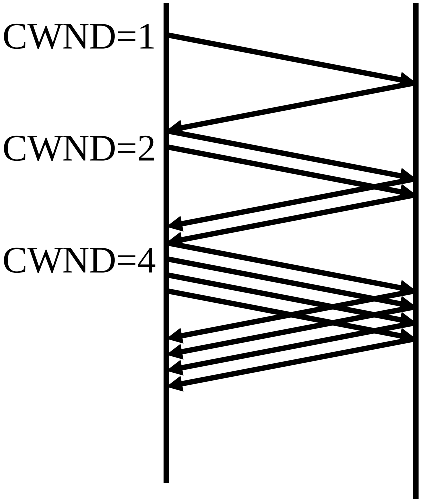
7. What is the length of each of the first six TCP segments?
8. What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?
9. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?
10. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 on page 247 in the text).
11. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

Optional Homework 3D

Try but do not submit.

A TCP entity opens a connection and uses slow start. Approximately how many round-trip times are required before TCP can send N segments.

Hint:



Acronyms

ACK	ACKnowledgement
AIMD	Additive increase and multiplicative decrease
ARQ	Automatic Repeat Request
CE	Congestion Experienced
CRC	Cyclic Redundancy Check
CWND	Congestion Window
CWR	Congestion Window Reduced
DA	Destination Address
DEC	Digital Equipment Corporation
DECbit	DEC's bit based congestion scheme
DevRTT	Deviation of RTT
DNS	Domain Name System
DP	Destination Port
ECE	Explicit Congestion Experienced
ECN	Explicit Congestion Notification
FIN	Final

Acronyms (Cont)

FTP	File Transfer Protocol
GBN	Go-Back N
HTTP	Hyper-Text Transfer Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISN	Initial Sequence Number
kB	Kilo-Byte
MSS	Maximum segment size
PBX	Private Branch Exchange
PSH	Push
RFC	Request for Comments
RM	Resource Management
RST	Reset
RTT	Round-Trip Time
SA	Source Address
SACK	Selective Acknolowledgement

Acronyms (Cont)

SMTP	Simple Mail Transfer Protocol
SP	Source Port
SSThresh	Slow Start Threshold
SYN	Synchronization
SYNACK	SYN Acknowledgement
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URG	Urgent
VCI	Virtual Circuit Identifiers

Scan This to Download These Slides

Raj Jain
http://rajjain.com

Related Modules

CSE 473s: Introduction to Computer Networks (Course Overview), http://www.cse.wustl.edu/~jain/cse473-16/ftp/i_0int.pdf

CSE473S: Introduction to Computer Networks (Fall 2016), http://www.cse.wustl.edu/~jain/cse473-16/index.html

Wireless and Mobile Networking (Spring 2016), http://www.cse.wustl.edu/~jain/cse574-16/index.html

CSE571S: Network Security (Fall 2014), http://www.cse.wustl.edu/~jain/cse571-14/index.html

Audio/Video Recordings and Podcasts of Professor Raj Jain's Lectures, https://www.youtube.com/channel/UCN4-5wzNP9-ruOzQMs-8NUw