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/cse473-11/

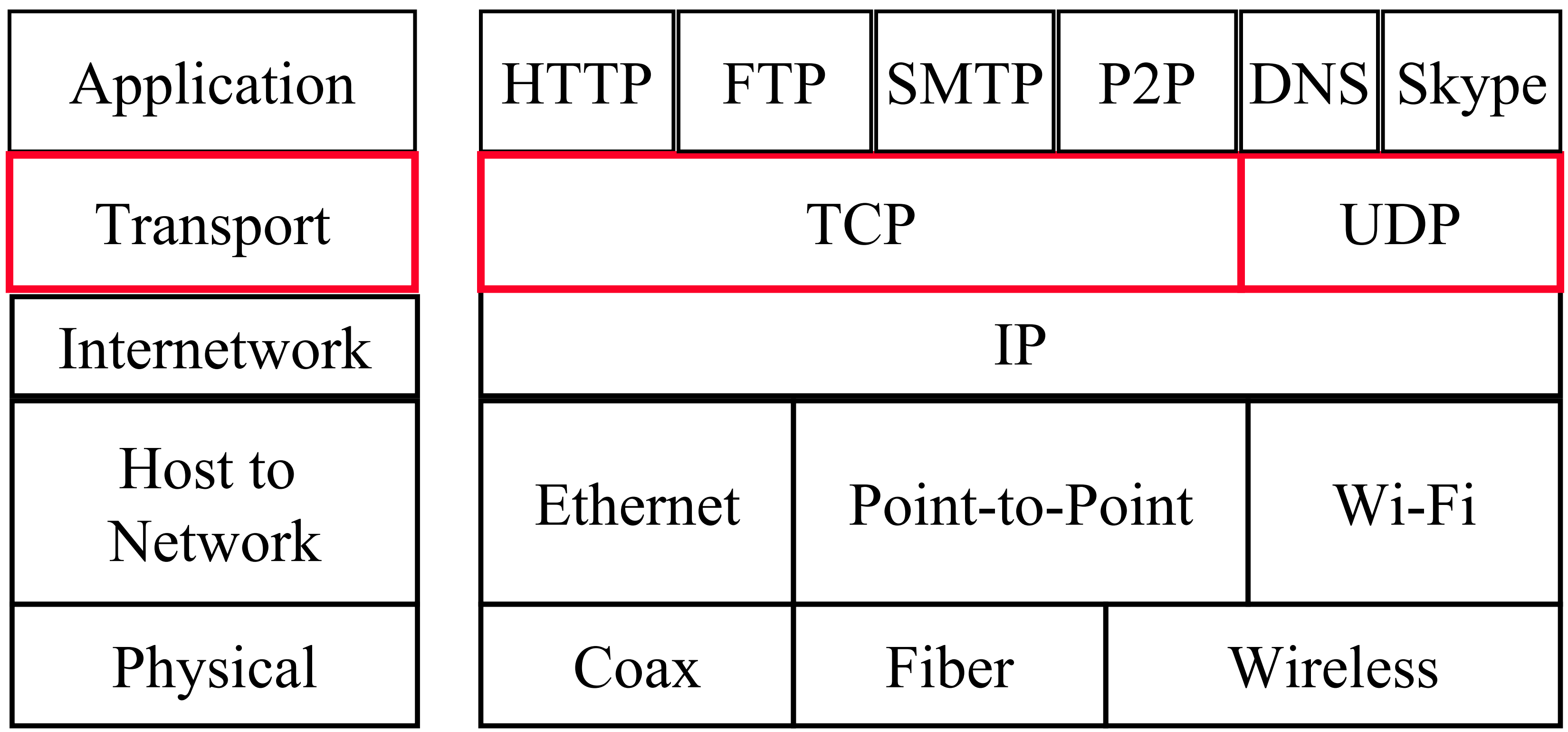
- Transport Layer Design Issues:
  - Multiplexing/Demultiplexing
  - Flow control
  - Error control
- UDP
- TCP
  - Header format, connection management, checksum
  - Slow Start Congestion Control
- **Note**: This class lecture is based on Chapter 3 of the textbook (Kurose and Ross) and the figures provided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

Transport Layer Design Issues

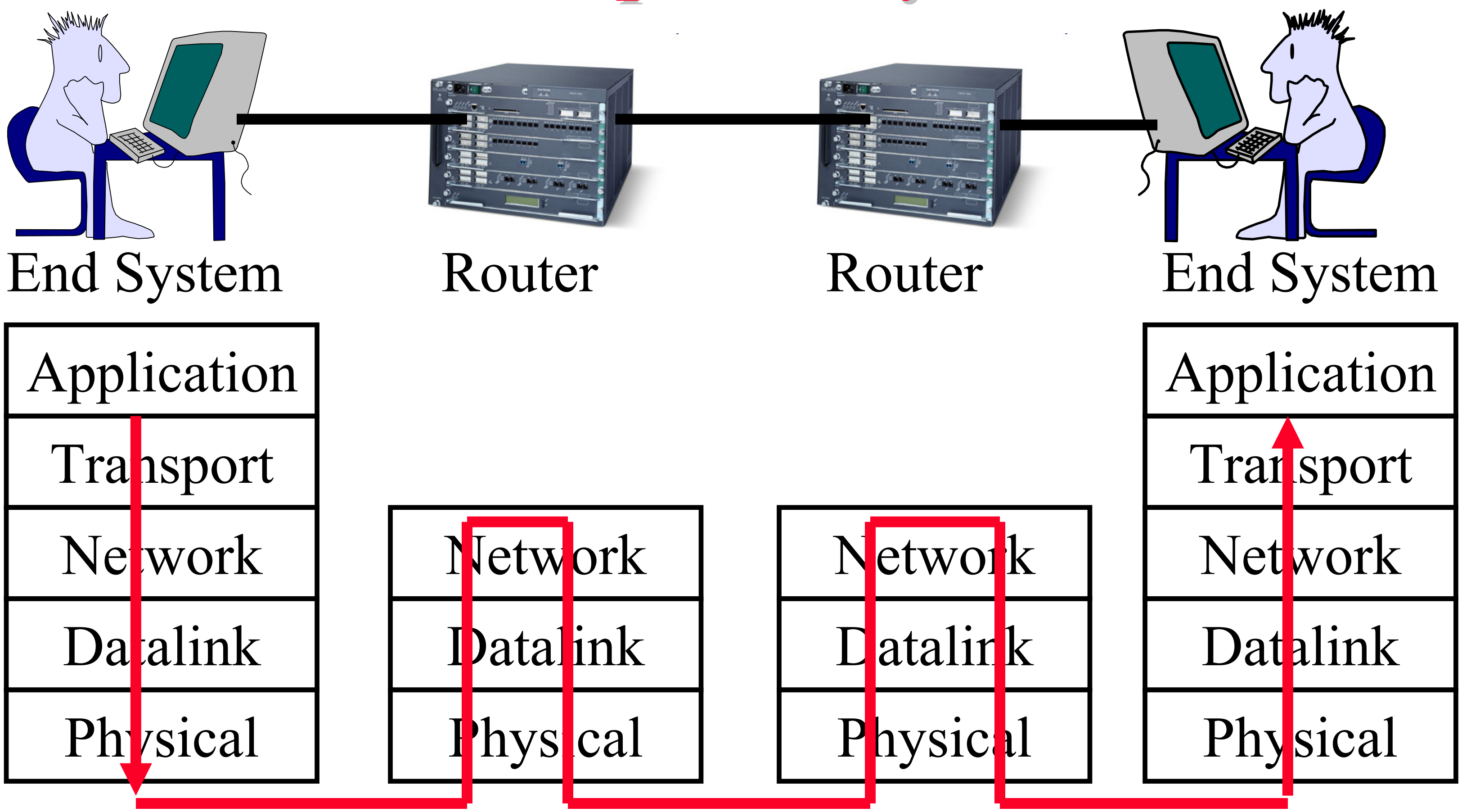
1. Transport Layer Functions
2. Multiplexing and Demultiplexing
3. Error Detection: Checksum
4. Flow Control
5. Efficiency Principle
6. Error Control: Retransmissions

Protocol Layers

- Top-Down approach



Transport Layer



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

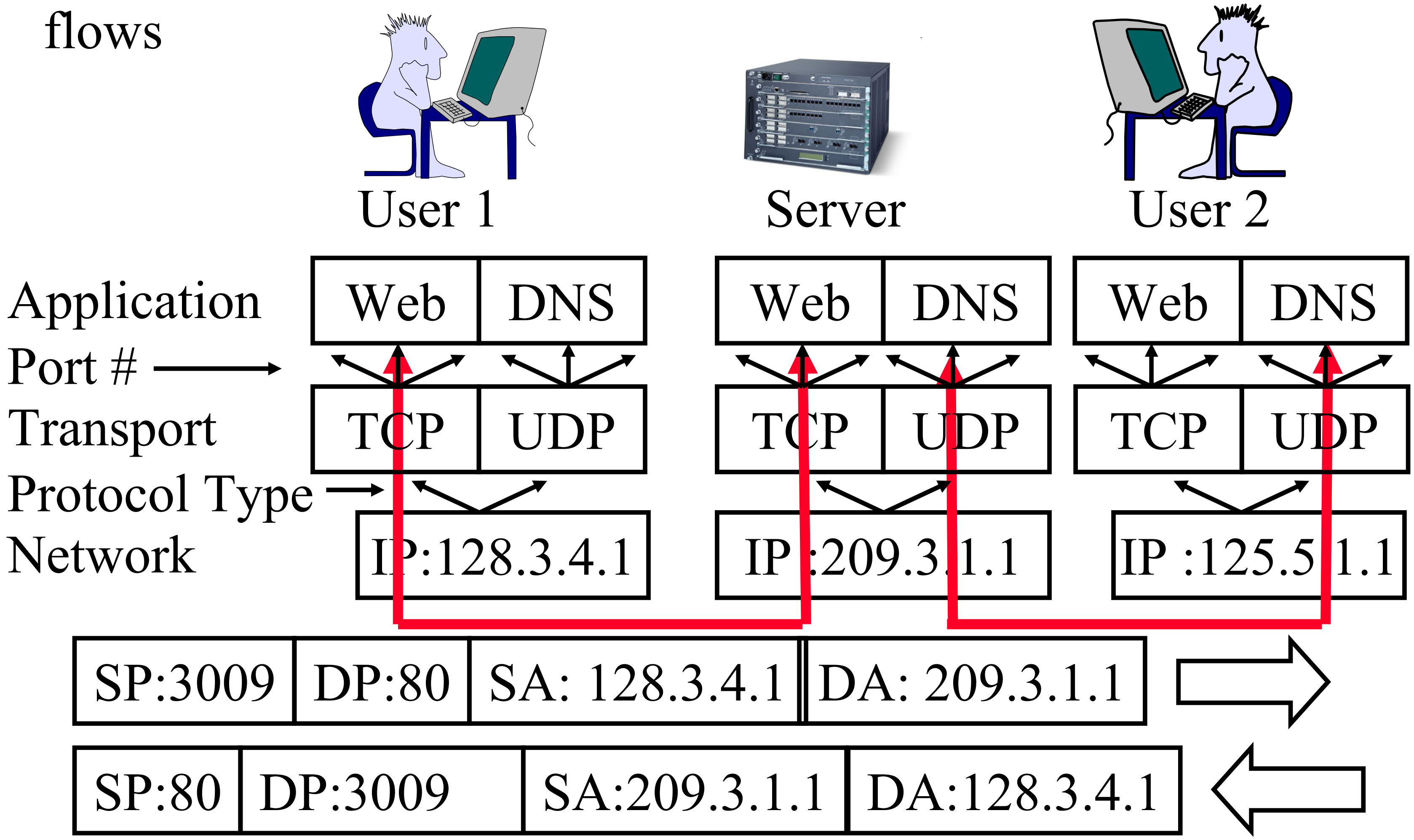
Transport Layer Functions

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

Multiplexing and Demultiplexing

- Transport Ports and Network addresses are used to separate flows



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

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 1+complement

- 1 = 0001 -1 = 1111
- 2 = 0010 -2 = 1110
- 3 = 0011 -3 = 1101

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

- 1 = 0001 -1 = 1110
- 2 = 0010 -2 = 1101
- 3 = 0011 -3 = 1100

2's Complement sum: Add with carry

1's complement sum: Add. Add the carry back to the sum

- 8+9 = 1000 + 1001 = 1 0001 => 0001 + 1 = 0010

Complement of 1's complement sum: 1101

Why: 1's complement sum is independent of the Endian-ness of the machines.

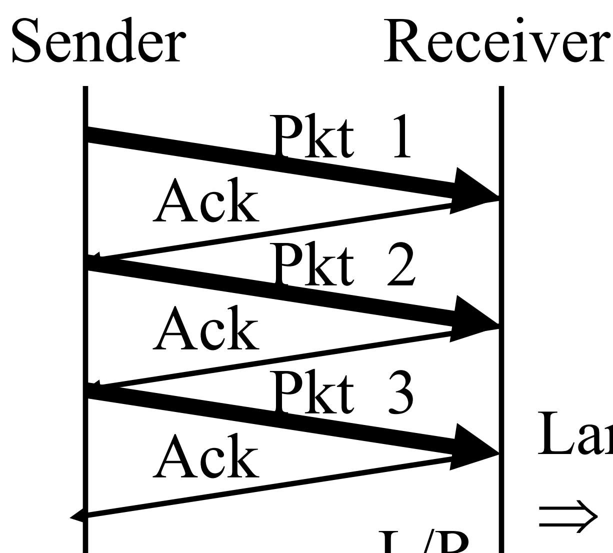
Little Endian = LSB is the left most bit.

Big Endian = MSB is the left most bit

Flow Control

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

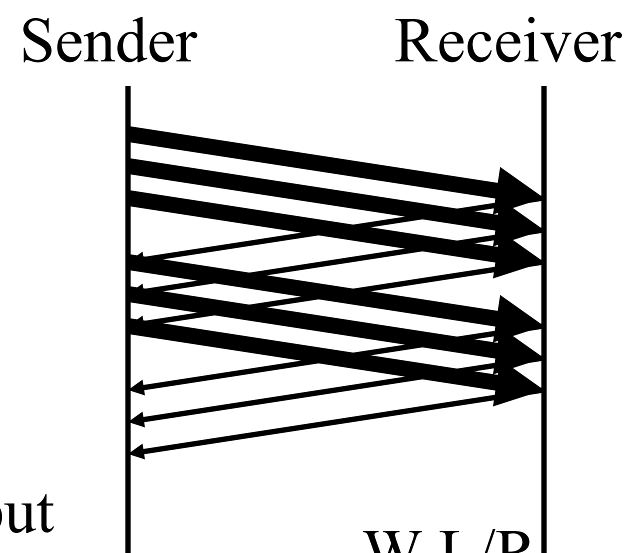
Stop and Wait Flow Control



Large RTT
⇒ Low Thruput

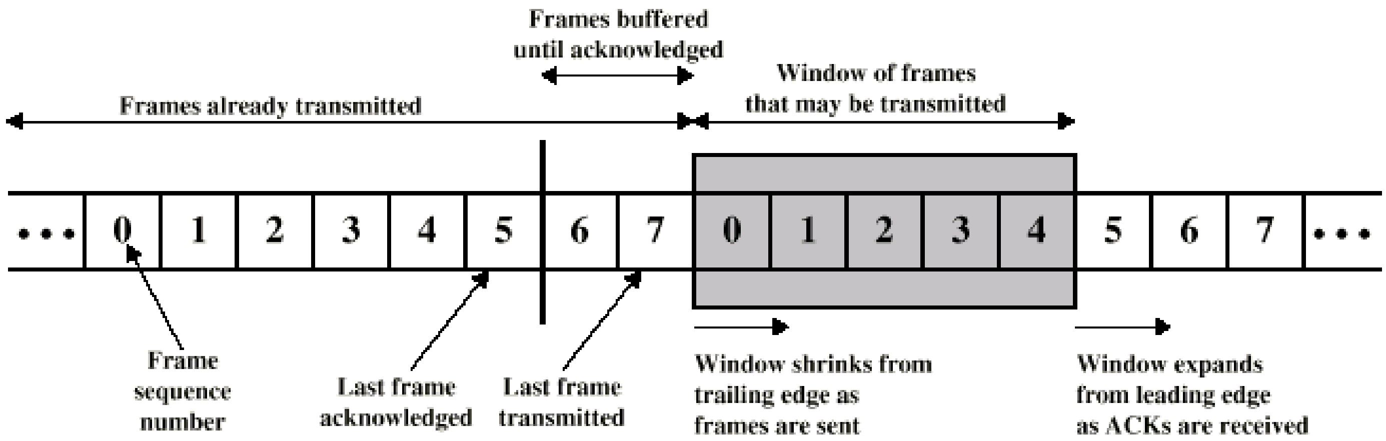
$$\text{Throughput} = \frac{L/R}{RTT+L/R}$$

Window Flow Control

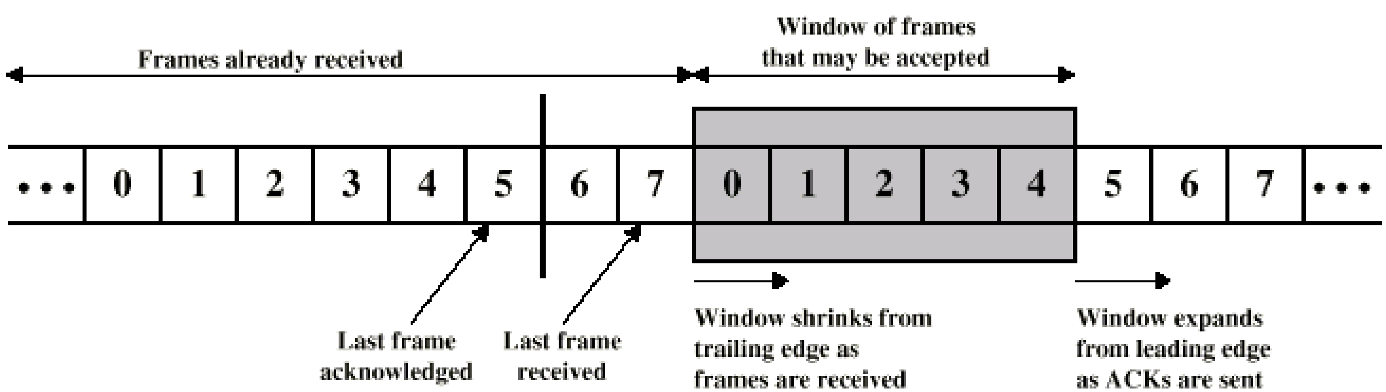


$$\text{Throughput} = \frac{W\ L/R}{RTT+L/R}$$

Sliding Window Diagram

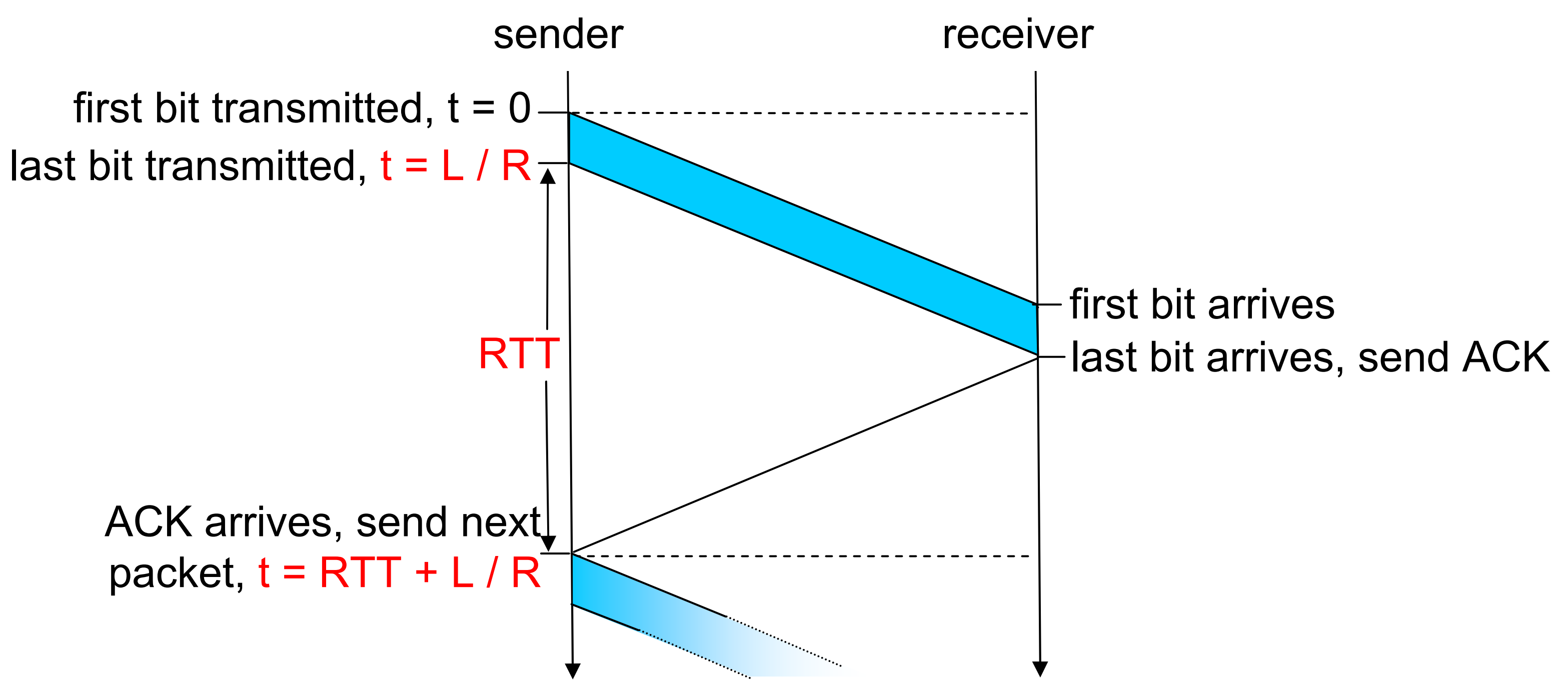


(a) Sender's perspective



(b) Receiver's perspective

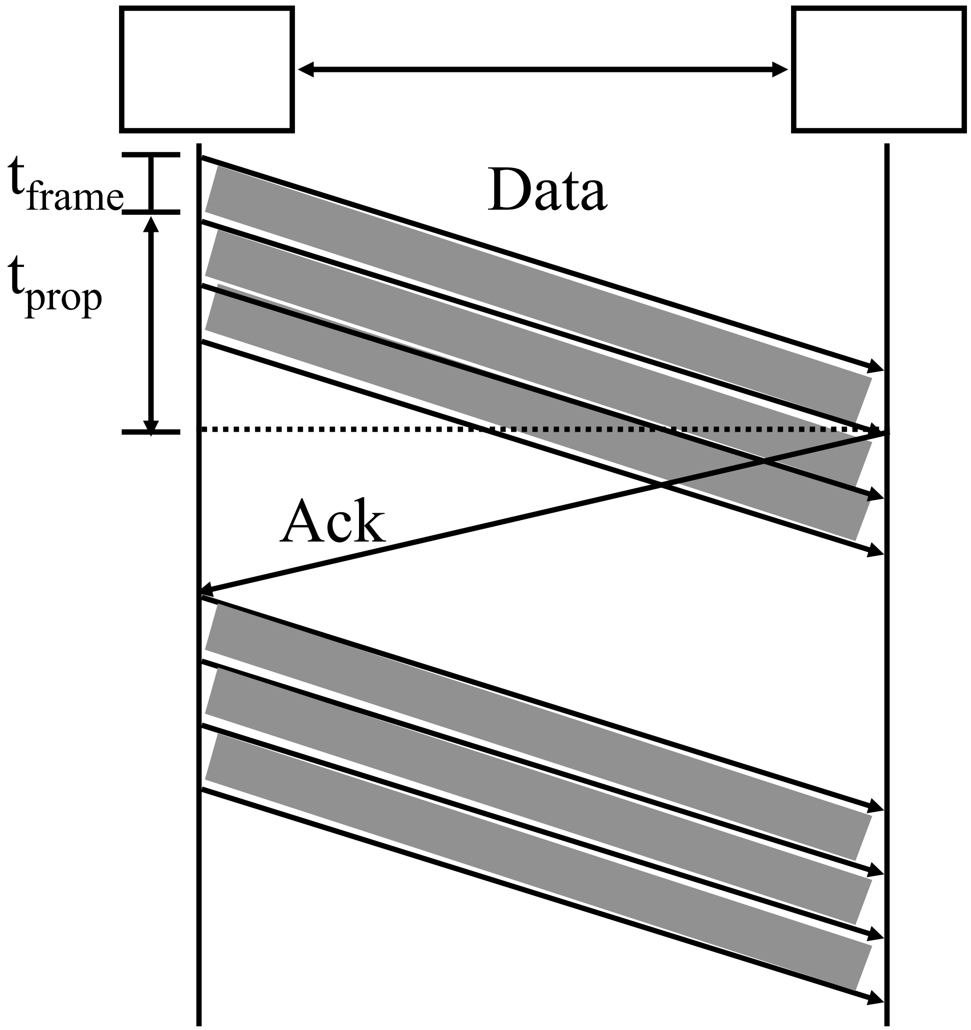
Stop and Wait Flow Control



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

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= 4/56 = 71 ms

$\alpha = t_{prop}/t_{frame} = 270/71 = 3.8$

$U = 1/(2\alpha+1) = 0.12$

- Short Link: 1 km = 5 μs,

  Rate=10 Mbps,

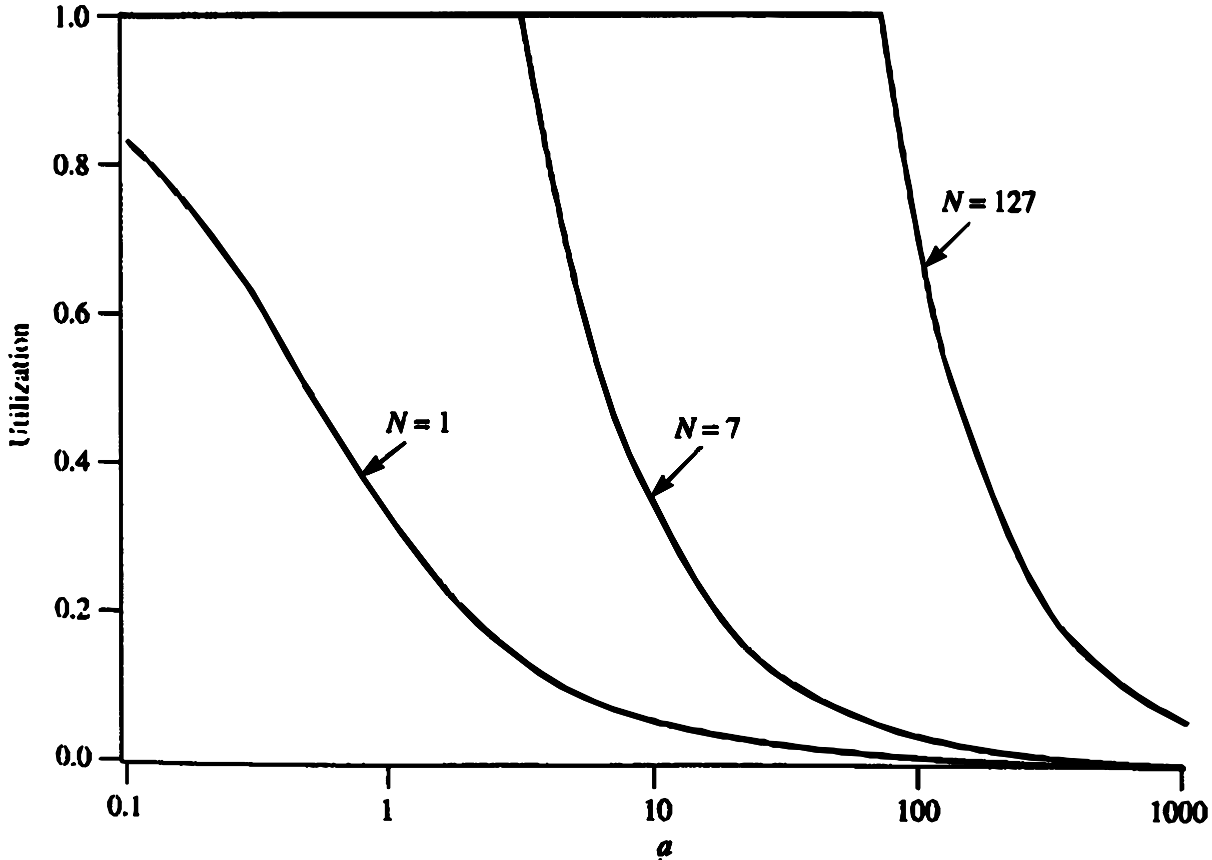
  Frame=500 bytes ⇒ t_{frame}= 4k/10M= 400 μs

  $\alpha=t_{prop}/t_{frame}=5/400=0.012 \Rightarrow U=1/(2\alpha+1)=0.98$

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

Effect of Window Size

U

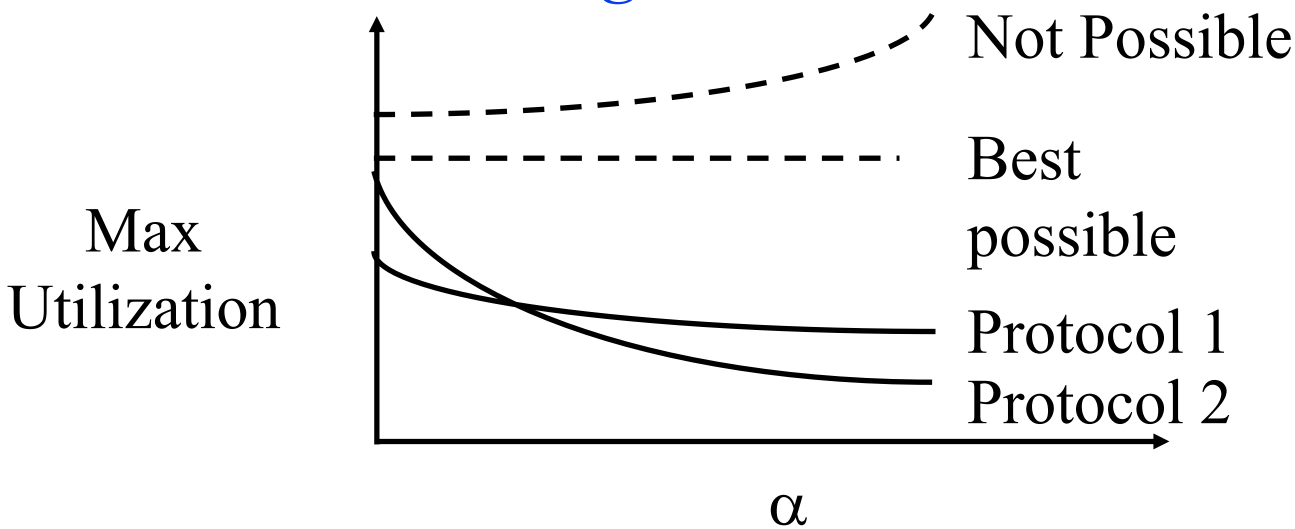


α

- Larger window is better for larger α

Efficiency Principle

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

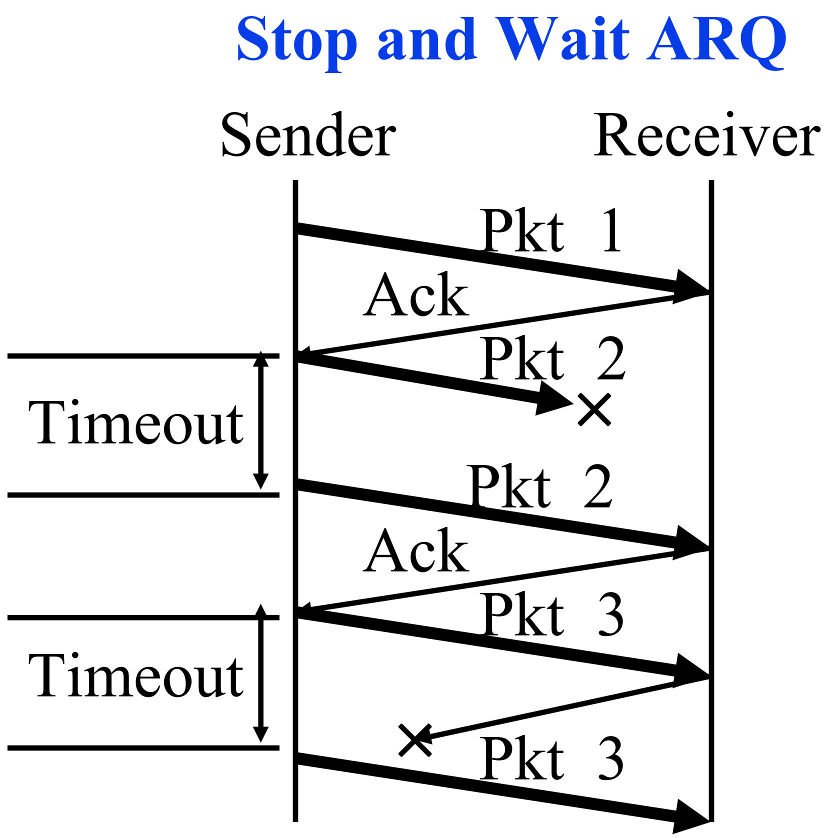


$$\alpha = \frac{t_{prop}}{t_{frame}} = \frac{\text{Distance/Speed of Signal}}{\text{Bits Transmitted /Bit rate}}$$

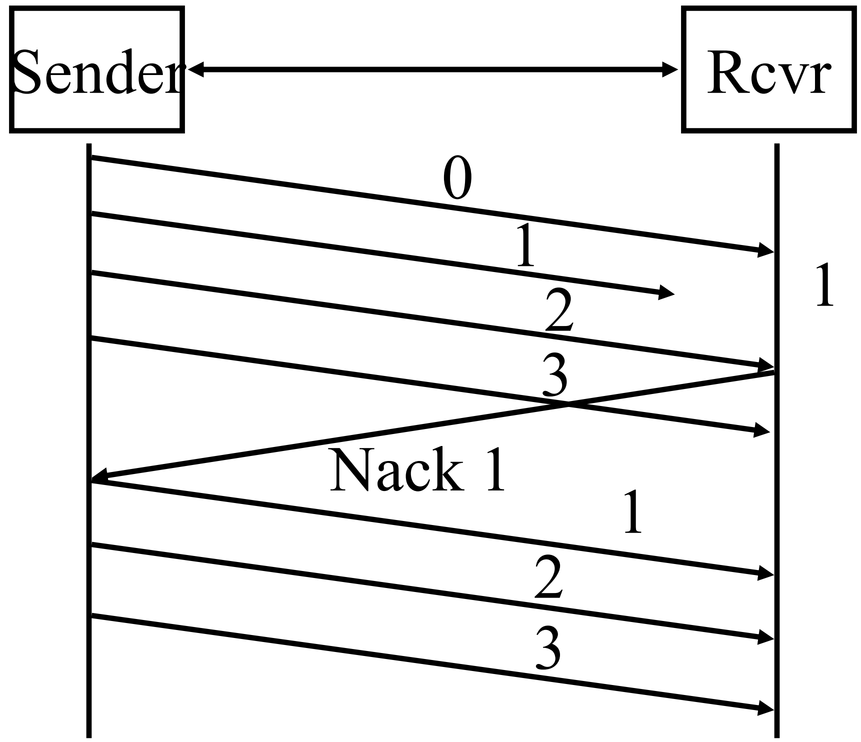
$$= \frac{\text{Distance} \times \text{Bit rate}}{\text{Bits Transmitted} \times \text{Speed of Signal}}$$

Error Control: Retransmissions

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

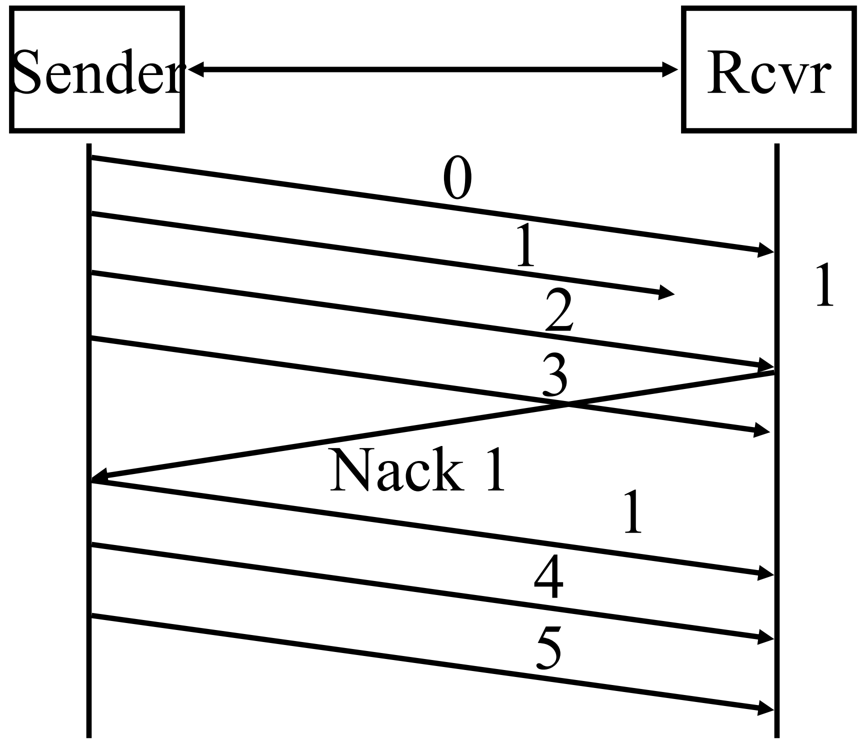


Go-Back-N ARQ



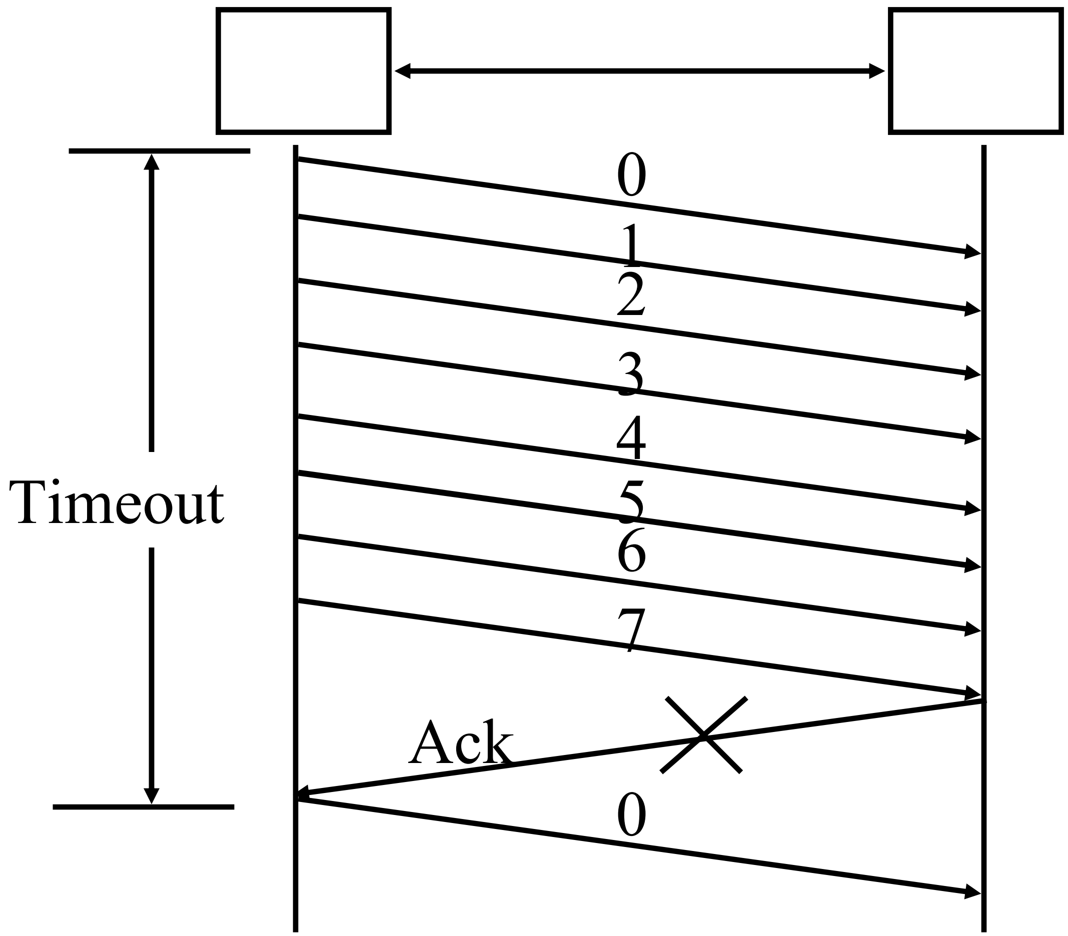
- Receiver does not cache out-of-order frames
- Sender has to ***go back*** and retransmit all frames after the lost frame

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**:

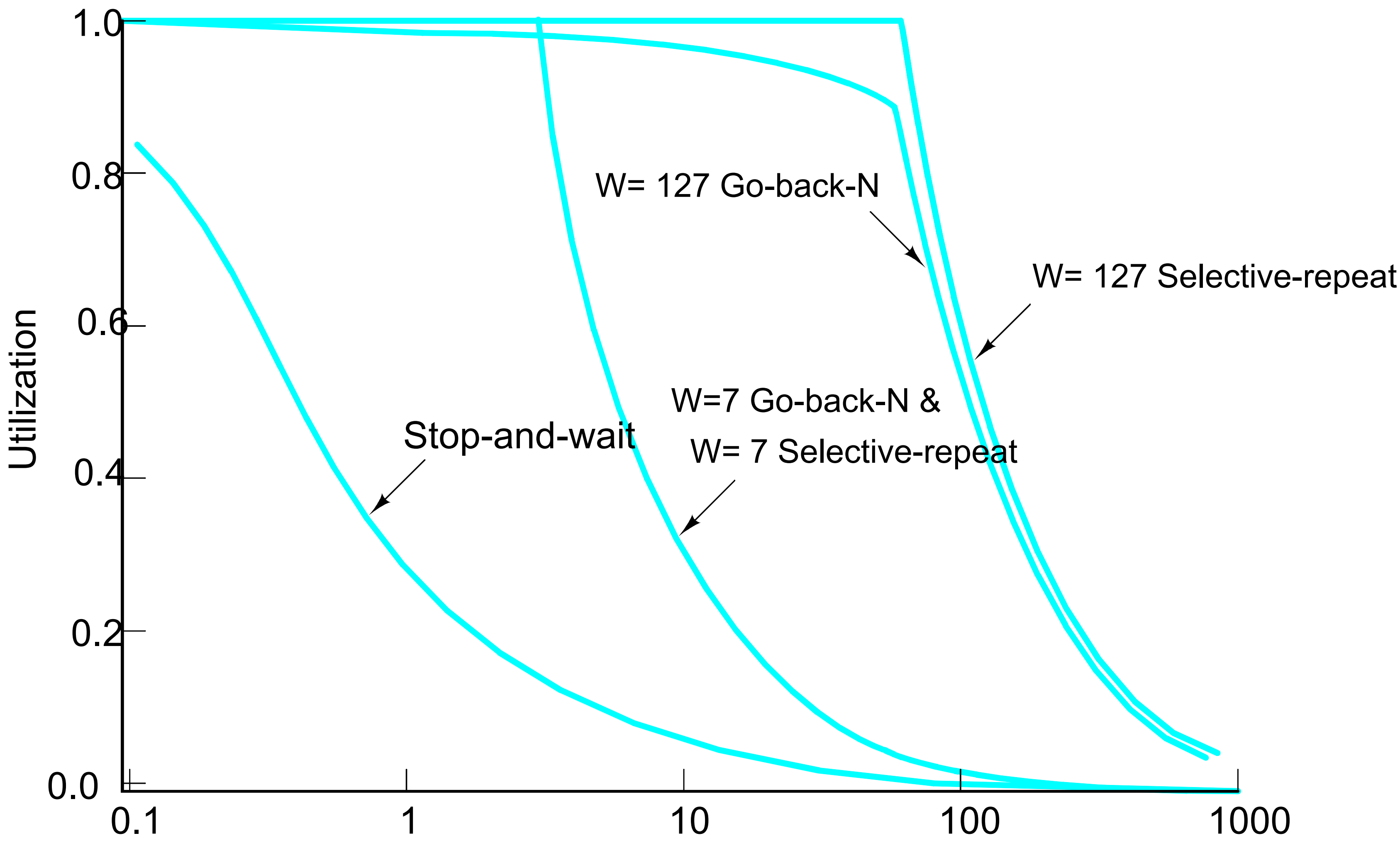
P = Probability of Loss

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

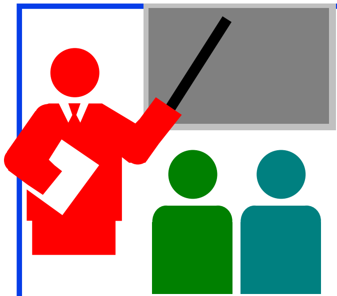
- **Selective Repeat ARQ**:

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

Performance Comparison



α More bps or longer distance →



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
   $\Rightarrow$ Larger α $\Rightarrow$ 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 stop and wait ARQ
   Only slightly better than go-back-N

Homework 3A

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?

UDP and TCP

1. User Datagram Protocol (UDP)
2. TCP Header Format, Options, Checksum
3. TCP Connection Management
4. Round Trip Time Estimation
5. Principles of Congestion Control
6. Slow Start Congestion Control

Transports

TCP	UDP
Reliable data transfer	Unreliable Data Transfer
Packet Sequence # required	Sequence # optional
Every packet is acked	Not Acked
Lost packets are retransmitted	No Retransmission
May cause long delay	Quick and Lossy
Connection-oriented service	Connection-less Service
Good for Reliable and delay-insenstive applications	Good for loss-tolerant and delay sensitive applications
Applications: email, http, ftp, Remote terminal access	Telephony, Streaming Multimedia

User Datagram Protocol (UDP)

- Connectionless end-to-end service
- No flow control. No error recovery (no acks)
- 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.
- 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	
16	16	16	16	← Size in bits

TCP Segment Format

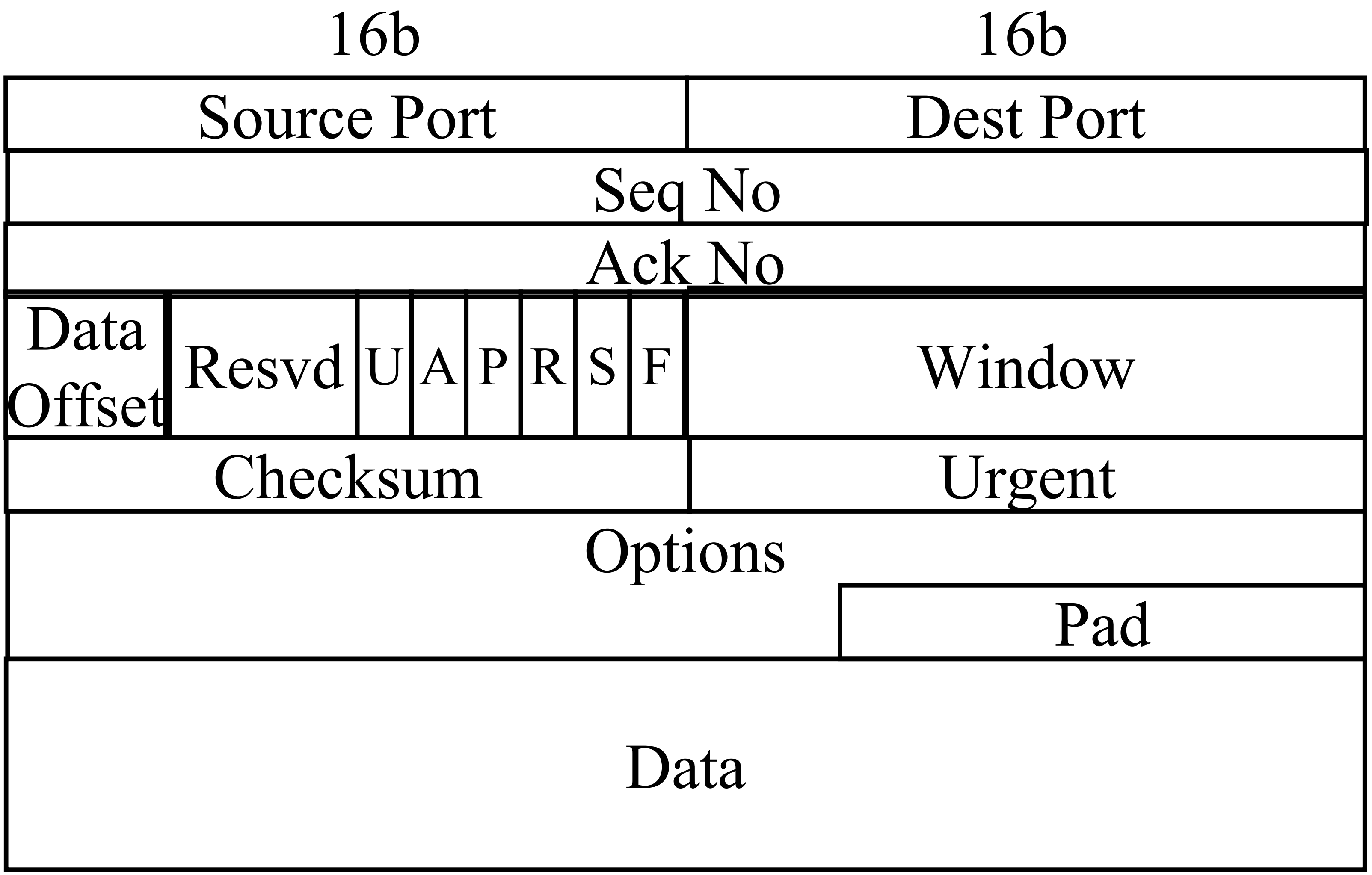
Source Port	Dest Port	Seq No	Ack No	Data Offset	Resvd	U	A	P	R	S	F
16	16	32	32	4	6	1	1	1	1	1	1

Window	Check-sum	Urgent	Options	Pad	Data
16	16	16	x	y	← Size in bits

TCP

- Transmission Control Protocol
- Key Services:
  - **Send**: Please send when convenient
  - **Data stream push**: Please send it all now, if possible.
  - **Urgent data signaling**: Destination TCP! please give this urgent data to the user
    (Urgent data is delivered in sequence. Push at the source should be explicit if needed.)
  - Note: Push has no effect on delivery.
    Urgent requests quick delivery

TCP Segment Format (Cont)



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.
- **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

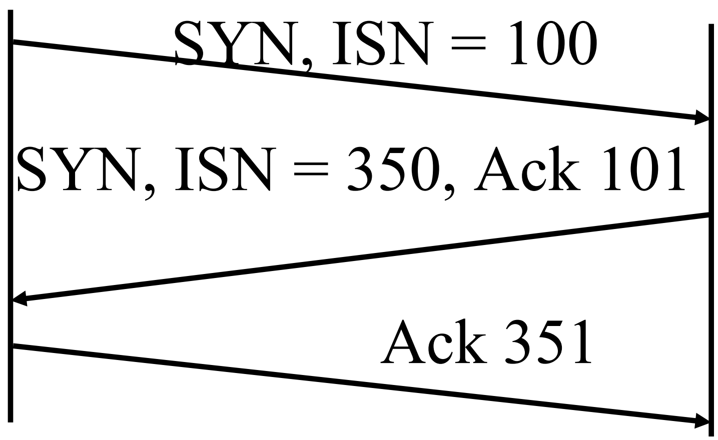
- Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the TCP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.
- Checksum field is filled with zeros initially
- TCP length (in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

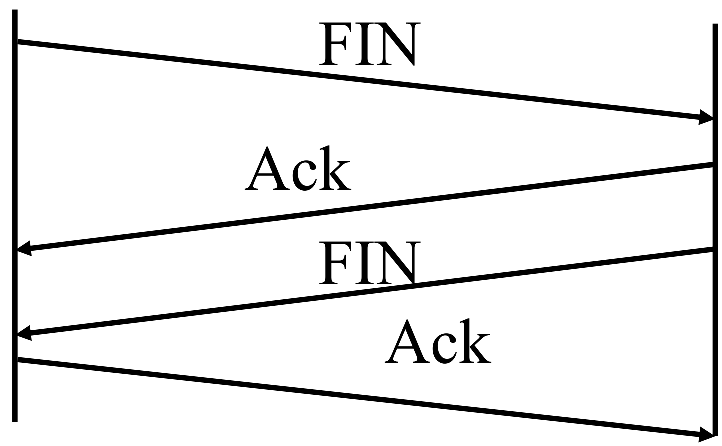
Source Adr	Dest. Adr	Zeros	Protocol	TCP Length
32	32	8	8	16

TCP Header	TCP data

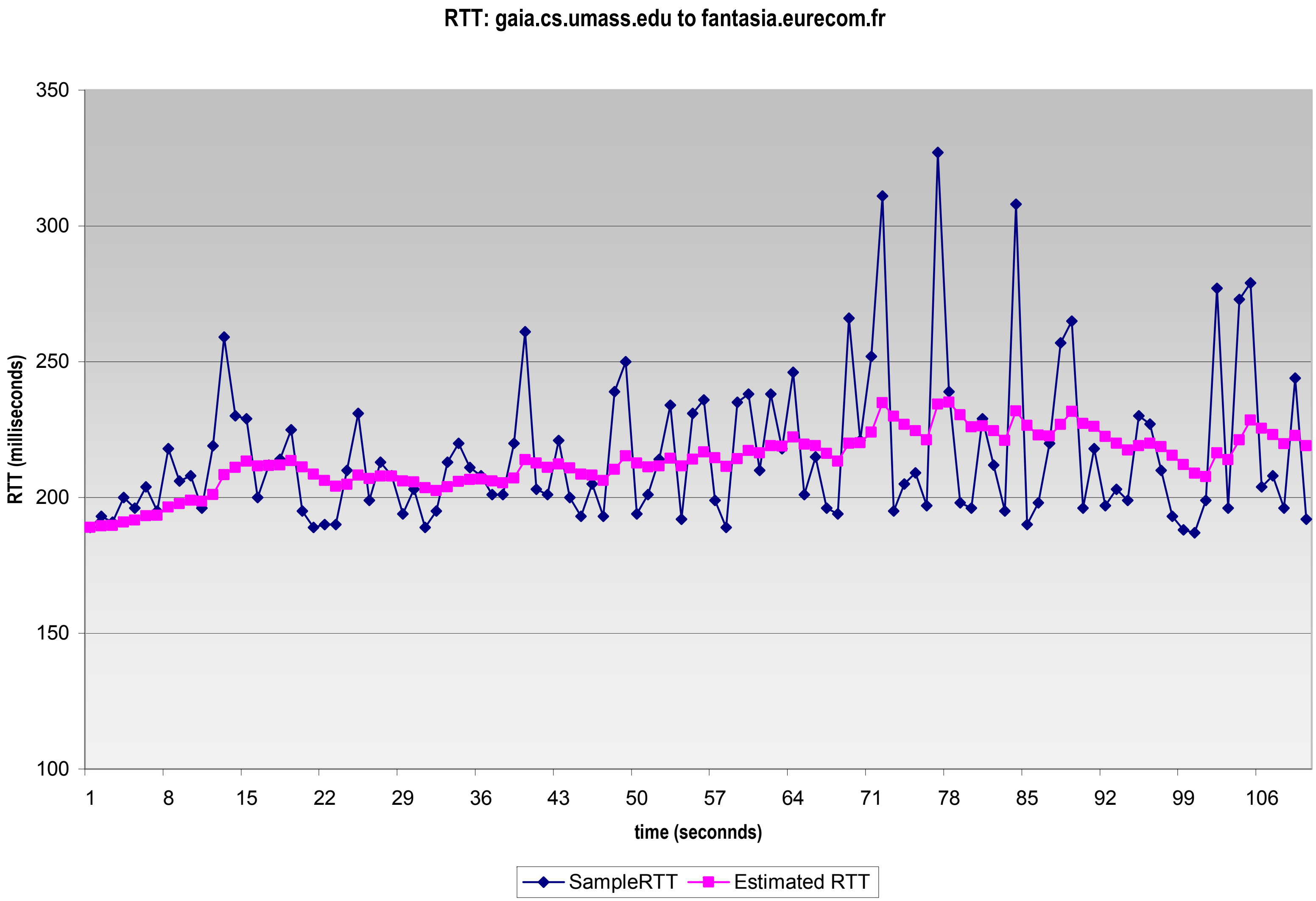
TCP Connection Management

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



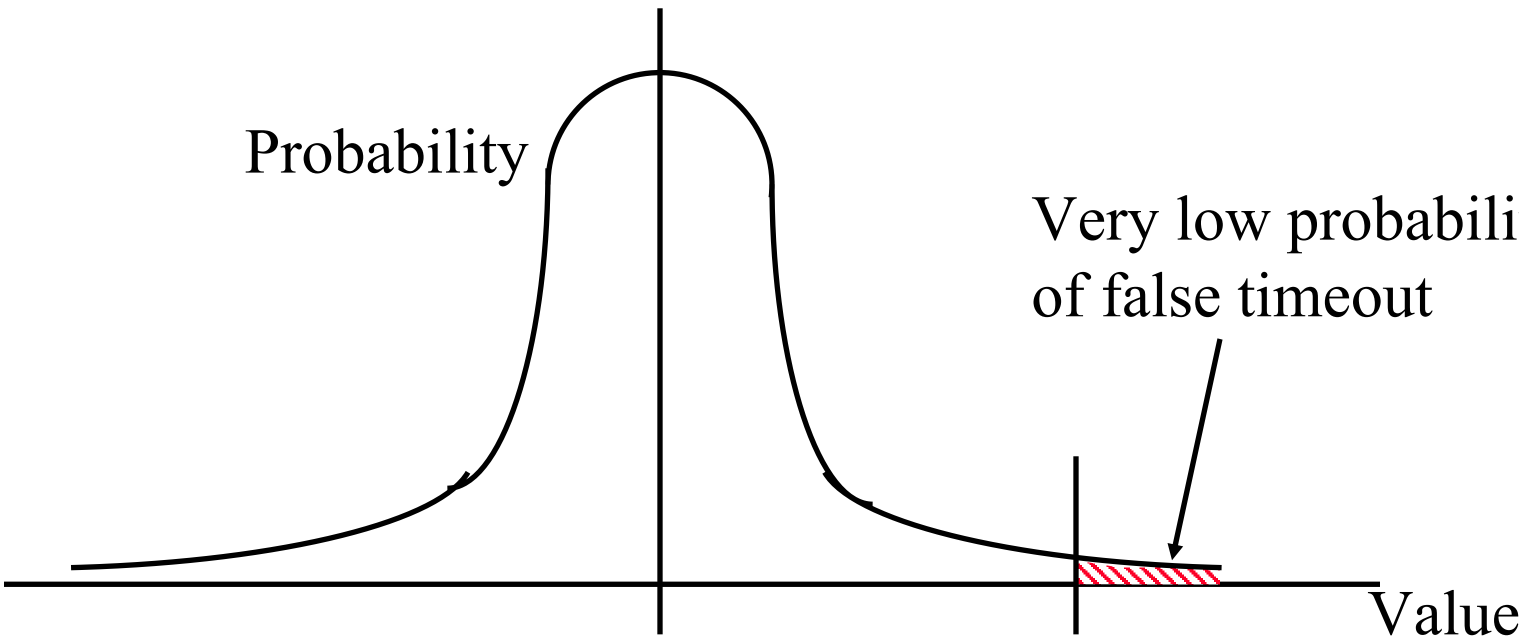


Example RTT estimation:



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

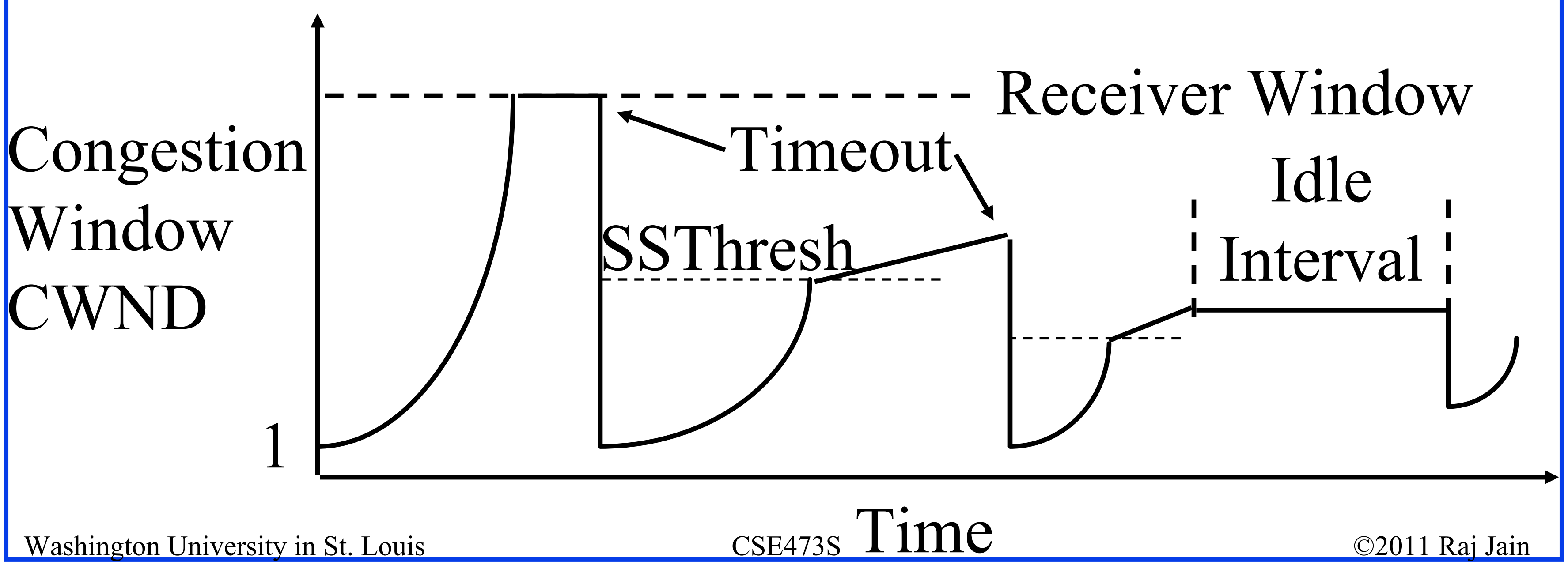


Slow Start Congestion Control

- 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 segment (one max segment size)
  ⇒ 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

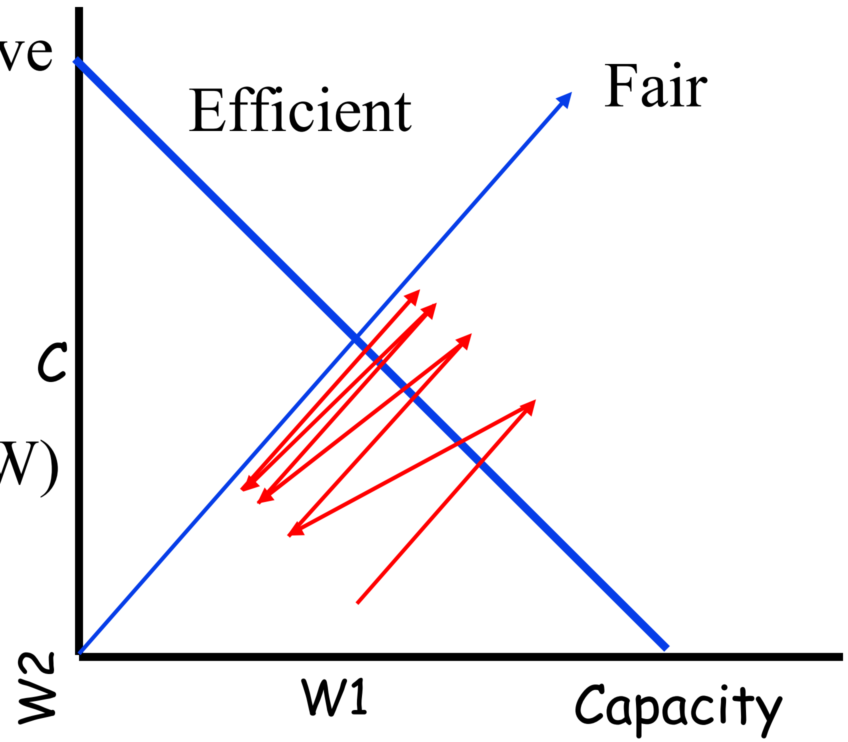


Slow Start (Cont)

- At the beginning, SSThresh = Receiver window
- After a long idle period (exceeding one round-trip time), reset the congestion window to one.
- Exponential growth phase is also known as "Slow start" phase
- 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+DW,W2+DW)
  ⇒ Linear increase (45° line)
- (W1,W2) to (kW1,kW2)
  ⇒ Multiplicative decrease
  (line through origin)

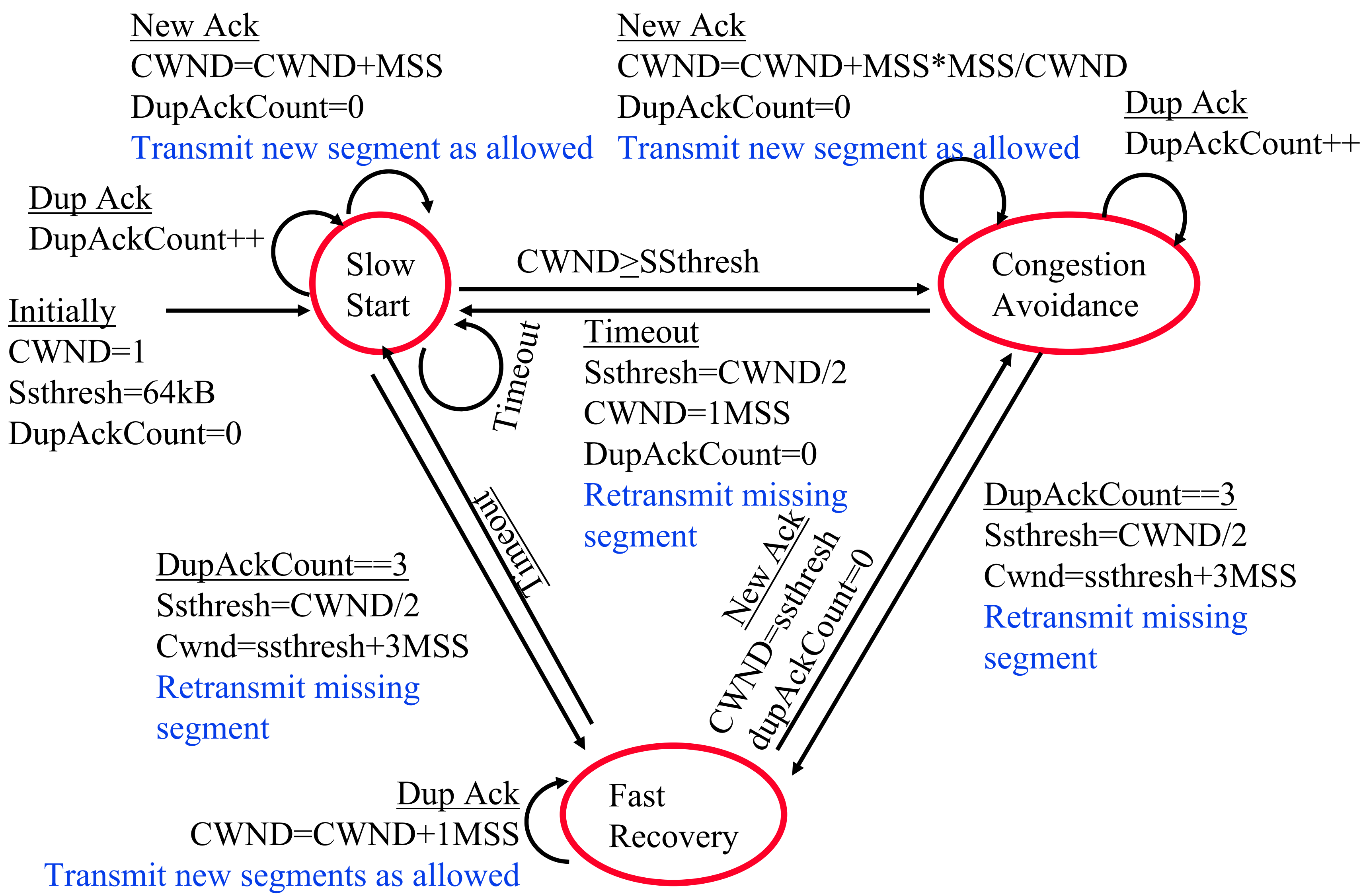


Ref: D. Chiu and Raj Jain, "**Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks**," Journal of Computer Networks and ISDN, Vol. 17, No. 1, June 1989, pp. 1-14, http://www.cse.wustl.edu/~jain/papers/cong_av.htm

Fast Recovery

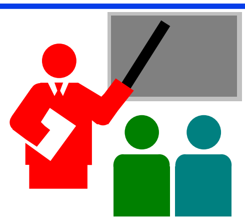
- Optional – implemented in TCP Reno
  (Earlier version was TCP Tahoe)
- Duplicate Ack indicates a lost/out-of-order segment
- On receiving 3 duplicate acks (4^{th} ack for the same segment):
  - Enter Fast Recovery mode
    - Retransmit missing segment
    - Set SSTHRESH=CWND/2
    - Set CWND=SSTHRESH+3 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



TCP Average Throughput

- Average Throughput = $\frac{1.22 \text{ MSS}}{\text{RTT} \sqrt{P}}$
- Here, P = Probability of Packet loss



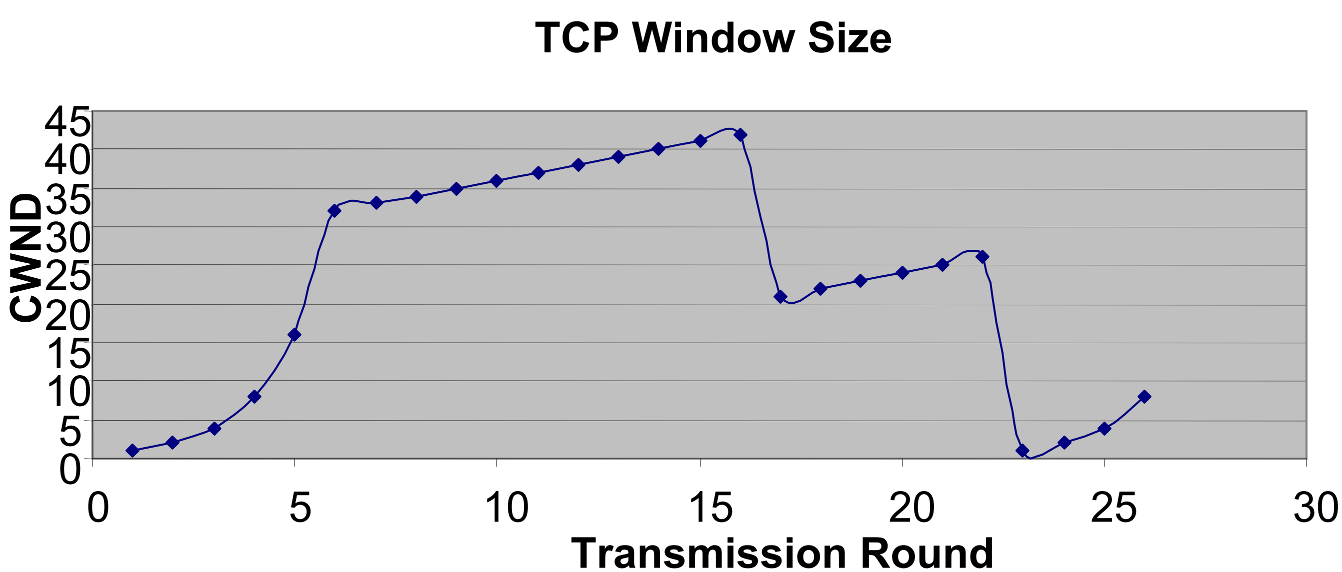
UDP and TCP: Summary

1. UDP provides flow multiplexing and optional checksum
2. Both UDP and TCP use port numbers for multiplexing
3. TCP provides reliable full-duplex connections.
4. TCP is stream based and has credit flow control
5. Slow-start congestion control works on timeout

Homework 3B

- Problem P37 on page 306 of the textbook:
- Consider Figure 3.58. 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 3B (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 3B (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
   $\Rightarrow$ Larger α $\Rightarrow$ 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.