

Transport Layer

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/cse473-09/>



- ❑ 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.



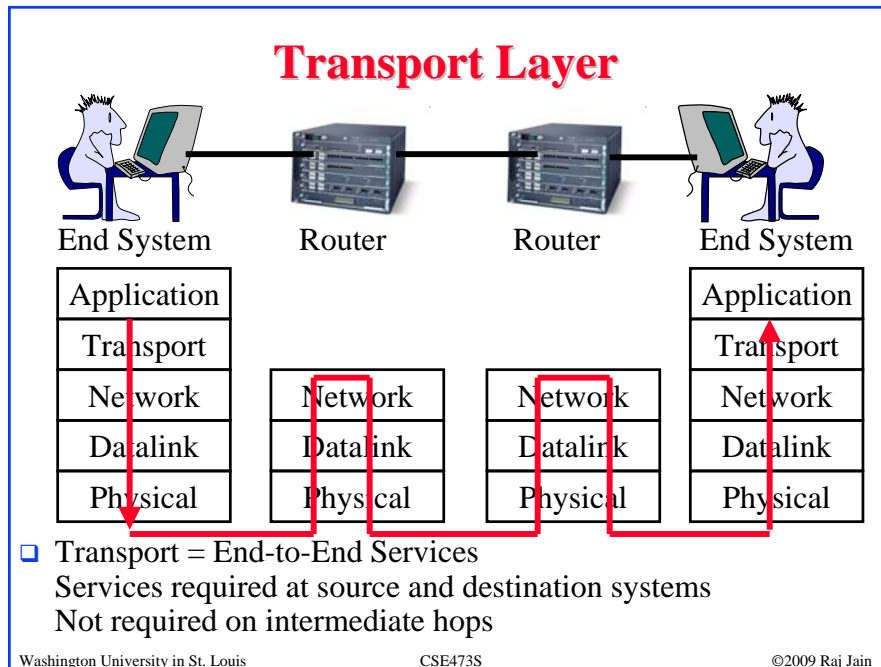
Transport Layer Design Issues

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

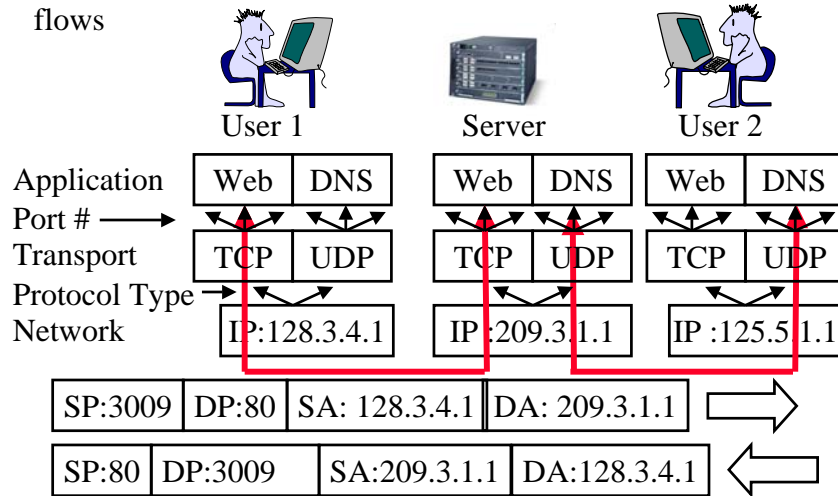
Application	HTTP	FTP	SMTP	P2P	DNS	Skype
Transport	TCP				UDP	
Internetwork	IP					
Host to Network	Ethernet	Point-to-Point		Wi-Fi		
Physical	Coax	Fiber	Wireless			



- ## 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
- Washington University in St. Louis CSE473S ©2009 Raj Jain

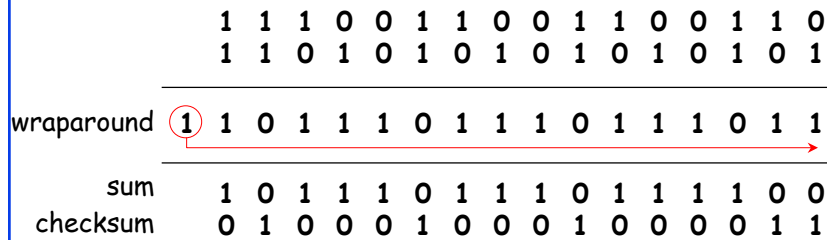
Multiplexing and Demultiplexing

- Transport Ports and Network addresses are used to separate flows



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



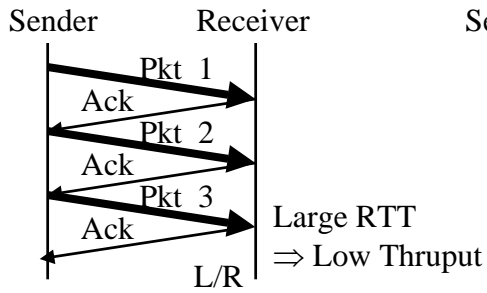
At receiver the checksum is zero.

Flow Control

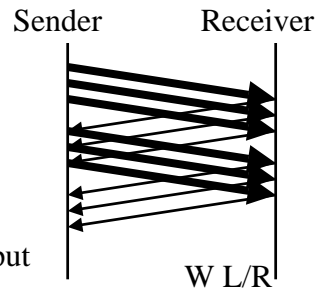


- Flow Control Goals:
 1. Sender does not flood the receiver,
 2. Maximize throughput

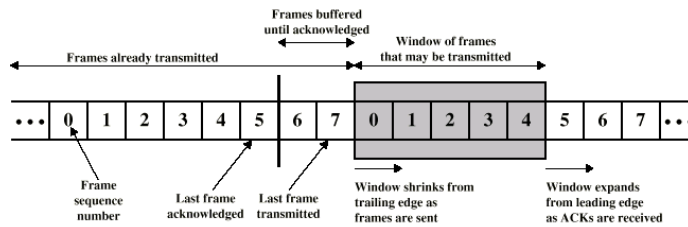
Stop and Wait Flow Control



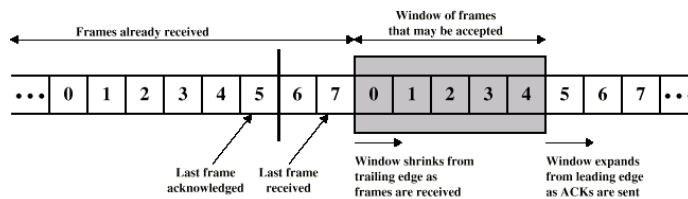
Window Flow Control



Sliding Window Diagram

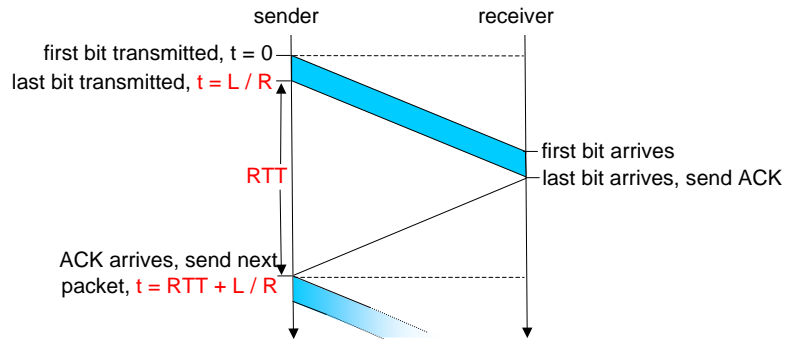


(a) Sender's perspective



(b) Receiver's perspective

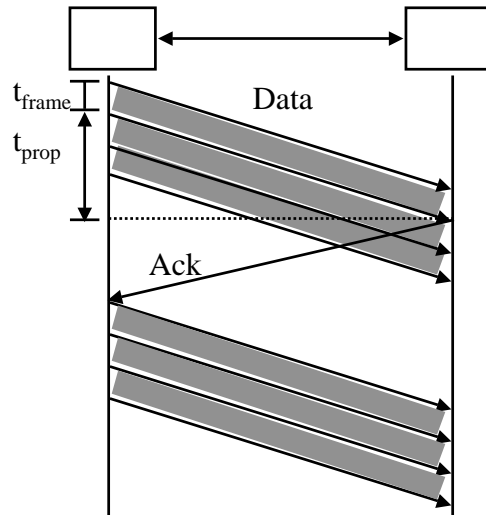
Stop and Wait Flow Control



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

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

Sliding Window Protocol Efficiency



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

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

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

$W=1 \Rightarrow$ 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 $\Rightarrow t_{\text{frame}} = L/R = 4/56 = 71$ ms

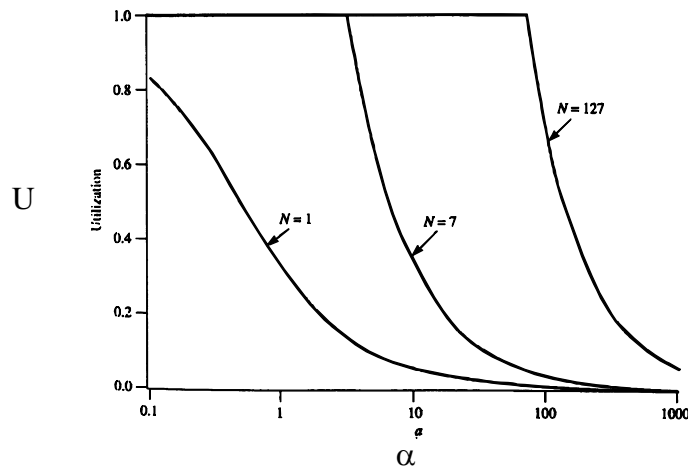
$\alpha = t_{\text{prop}}/t_{\text{frame}} = 270/71 = 3.8$

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

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

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

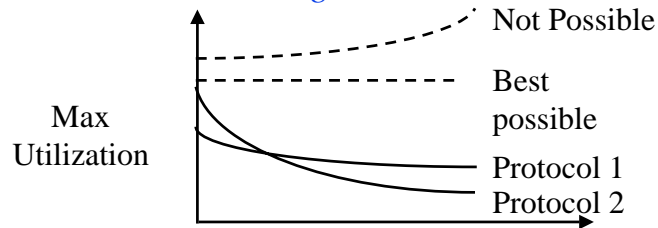
Effect of Window Size



- Larger window is better for larger α

Efficiency Principle

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



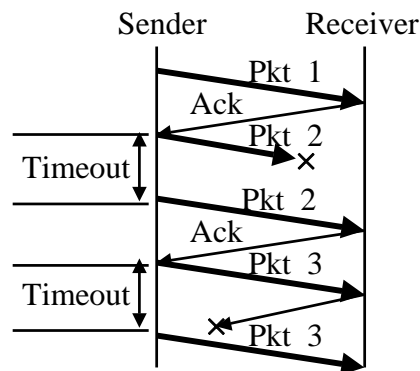
$$\alpha = \frac{t_{\text{prop}}}{t_{\text{frame}}} = \frac{\text{Distance/Speed of Signal}}{\text{Bits Transmitted /Bit rate}}$$

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

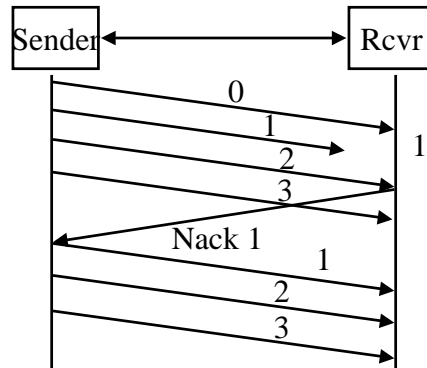
Error Control: Retransmissions

- Retransmit lost packets \Rightarrow Automatic Repeat reQuest (ARQ)

Stop and Wait ARQ

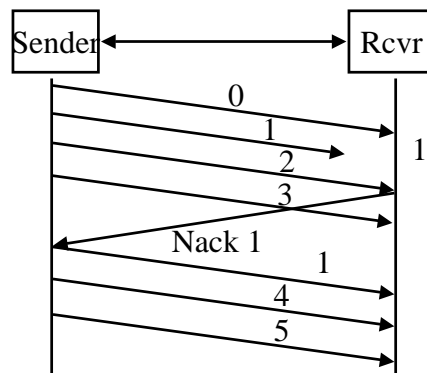


Go-Back-N ARQ



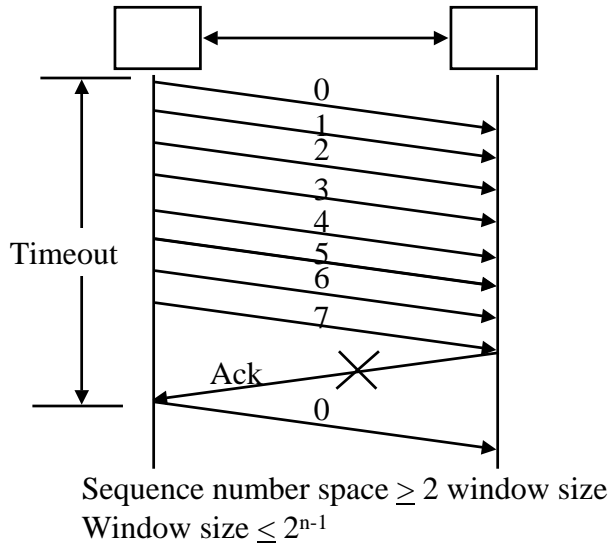
- ❑ Receiver does not cache out-of-order frames
- ❑ Sender has to *go back* and retransmit all frames after the lost frame

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



Performance: Maximum Utilization

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

□ **Window Flow Control:**

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

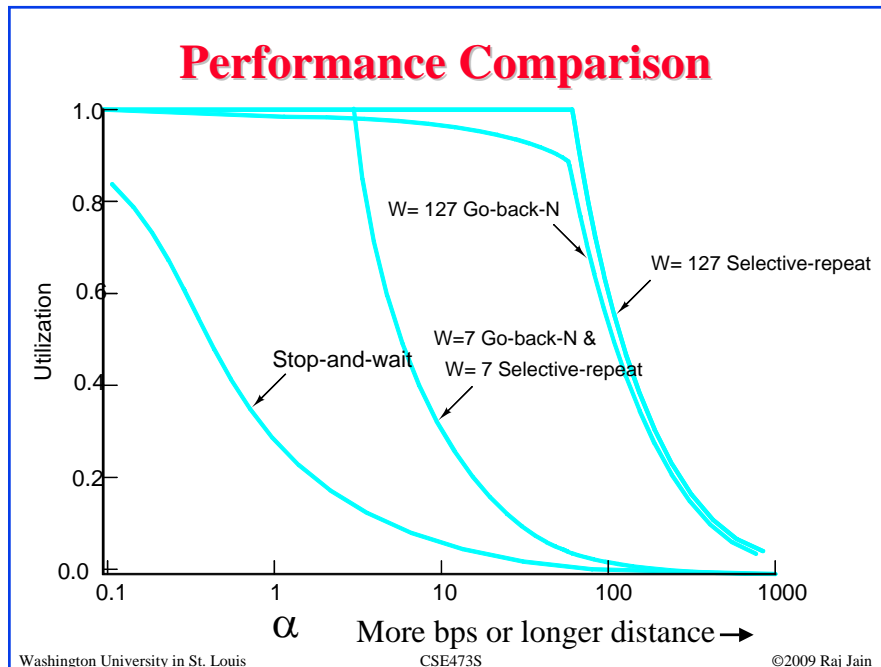
□ **Stop and Wait ARQ:** $U = (1-P)/(1+2\alpha)$

□ **Go-back-N ARQ:** $P = \text{Probability of Loss}$

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

□ **Selective Repeat ARQ:**

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



3-21



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 $\alpha \Rightarrow$ 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 inside 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.



UDP and TCP

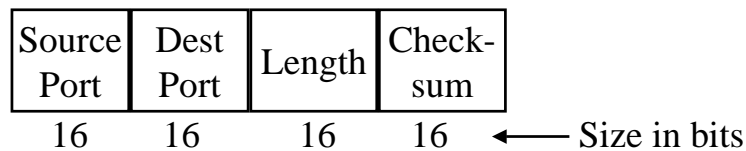
1. User Datagram Protocol (UDP)
2. TCP Header Format, Options, Checksum
3. TCP Connection Management
4. Round Trip Time Estimation
5. Principles of Congestion Control
6. Slow Start Congestion Control

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-insensitive applications	Good for loss-tolerant and delay sensitive applications
Applications: email, http, ftp, Remote terminal access	Telephony, Streaming Multimedia

User Datagram Protocol (UDP)

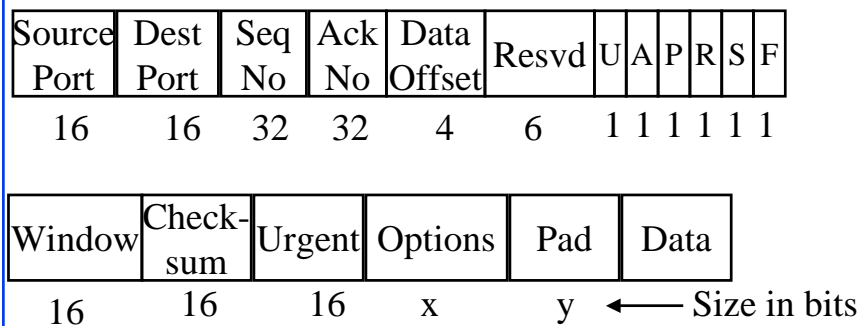
- ❑ Connectionless end-to-end service
- ❑ No flow control. No error recovery (no acks)
- ❑ Provides multiplexing via ports
- ❑ Error detection (Checksum) optional. Applies to pseudo-header (same as TCP) and UDP segment. If not used, it is set to zero.
- ❑ Used by network management, DNS, Streamed multimedia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



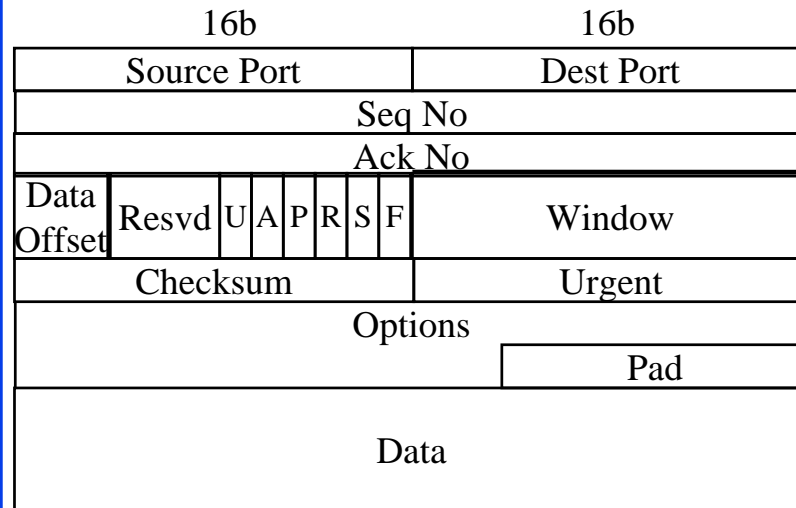
TCP

- ❑ 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



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



- **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

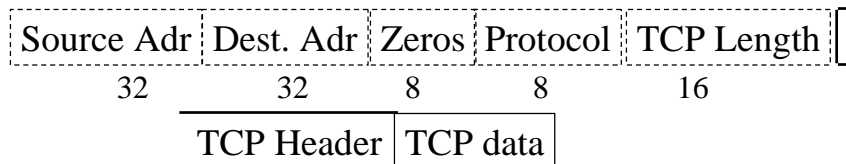
TCP Options

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

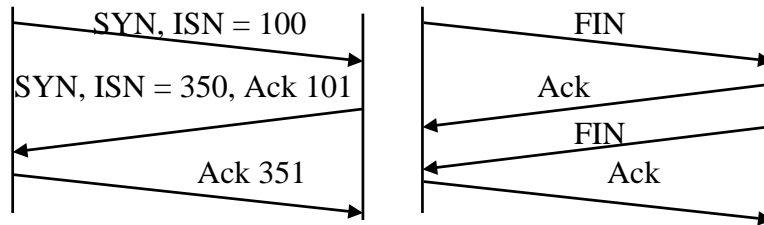
TCP Checksum

- ❑ 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.



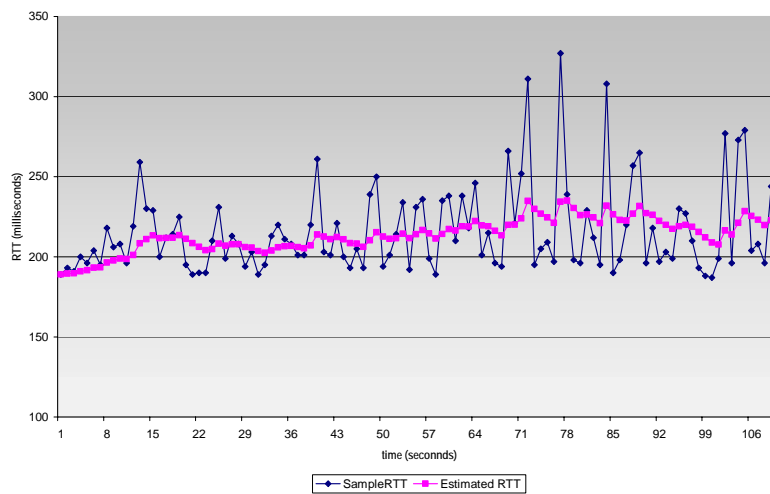
TCP Connection Management

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



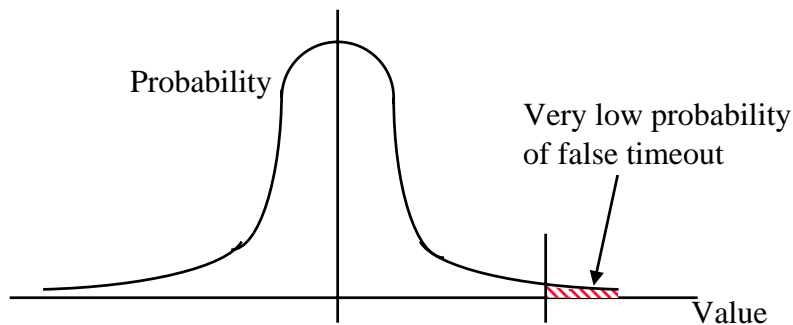
Example RTT estimation:

RTT: gaia.cs.umass.edu to fantasia.eurecom.fr



Round Trip Time Estimation

- ❑ Measured round trip time (SampleRTT) is very random.
- ❑ $\text{EstimatedRTT} = (1 - \alpha)\text{EstimatedRTT} + \alpha \text{ SampleRTT}$
- ❑ $\text{DevRTT} = (1 - \beta)\text{DevRTT} + \beta |\text{SampleRTT} - \text{EstimatedRTT}|$
- ❑ $\text{TimeoutInterval} = \text{EstimatedRTT} + 4 \text{ DevRTT}$



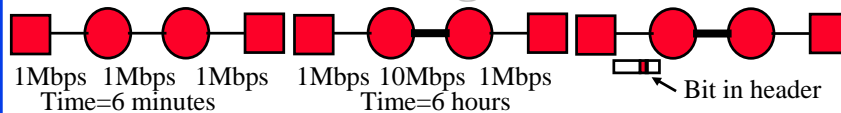
Washington University in St. Louis

CSE473S

©2009 Raj Jain

3-37

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 \Rightarrow Congestion
 - \Rightarrow 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.
 - \Rightarrow Slow-start based on Timeout and AIMD [Van Jacobson 1988]

Washington University in St. Louis

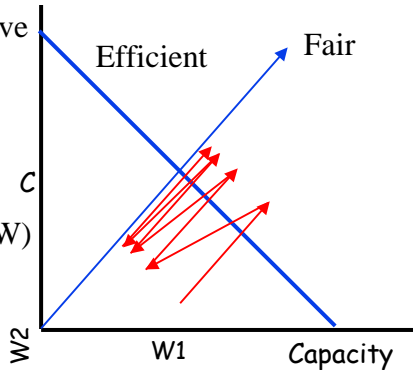
CSE473S

©2009 Raj Jain

3-38

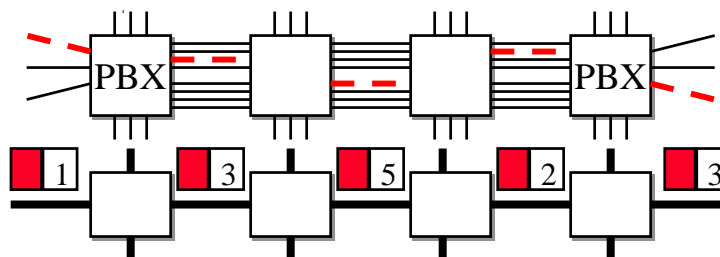
AIMD Principle

- Additive Increase, Multiplicative Decrease
- $W1+W2 = \text{Capacity}$
 \Rightarrow Efficiency,
 $W1=W2 \Rightarrow$ Fairness
- $(W1, W2)$ to $(W1+DW, W2+DW)$
 \Rightarrow Linear increase (45° line)
- $(W1, W2)$ to $(kW1, kW2)$
 \Rightarrow 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

ATM Networks



- Asynchronous transfer mode
- Uses fixed size 53-byte **cells**
- **Connection Oriented Network Layer**
 \Rightarrow Connection setup before data transfer
- Cells contain **Virtual Circuit Identifiers (VCI)**
- Switch forwarding tables contain:
 Input Interface + VCI \rightarrow Output Interface + New VCI

ATM ABR Service



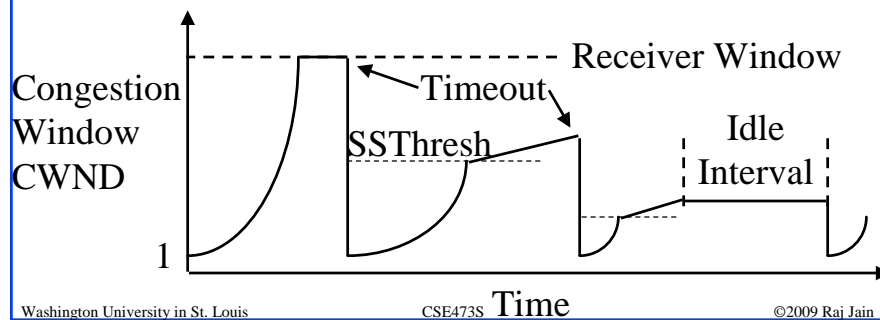
- Explicit Rate Feedback for congestion control
 - ⇒ Available bit rate (ABR) service
 - Every 32nd cell is a Resource Management (RM) cell. Switches put the rate in the cell. Destinations return the cell to source.
- Ref: Anna Charny, David D. Clark, Raj Jain, "**Congestion Control with Explicit Rate Indication**," ATM Forum/94-0692, July 1994, http://www.cse.wustl.edu/~jain/atmf/af_mit2.htm

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

Slow Start (Cont)

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



3-43

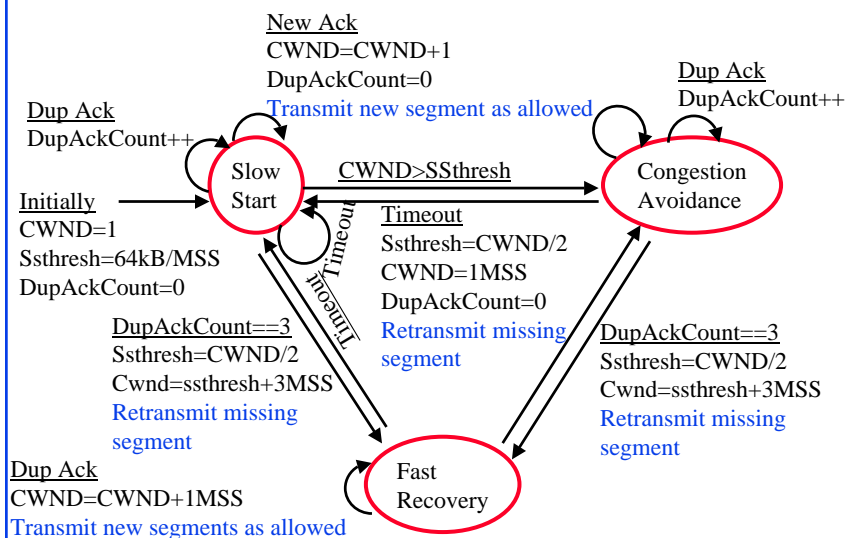
Slow Start (Cont)

- At the beginning, $SSThresh = \text{Receiver window}$
- 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”

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:
 - ❑ Enter Fast Recovery mode
 - ❑ Retransmit missing segment
 - ❑ Set $SSTHRESH = CWND/2$
 - ❑ Set $CWND = SSTHRESH + 3 \text{ MSS}$
 - ❑ Every subsequent duplicate ack: $CWND = CWND + 1 \text{ MSS}$

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

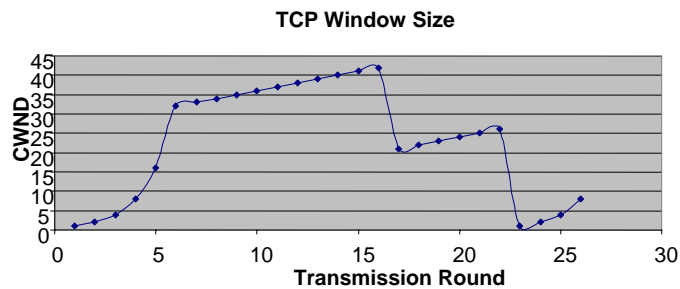


UDP and TCP: Summary

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.



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 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.