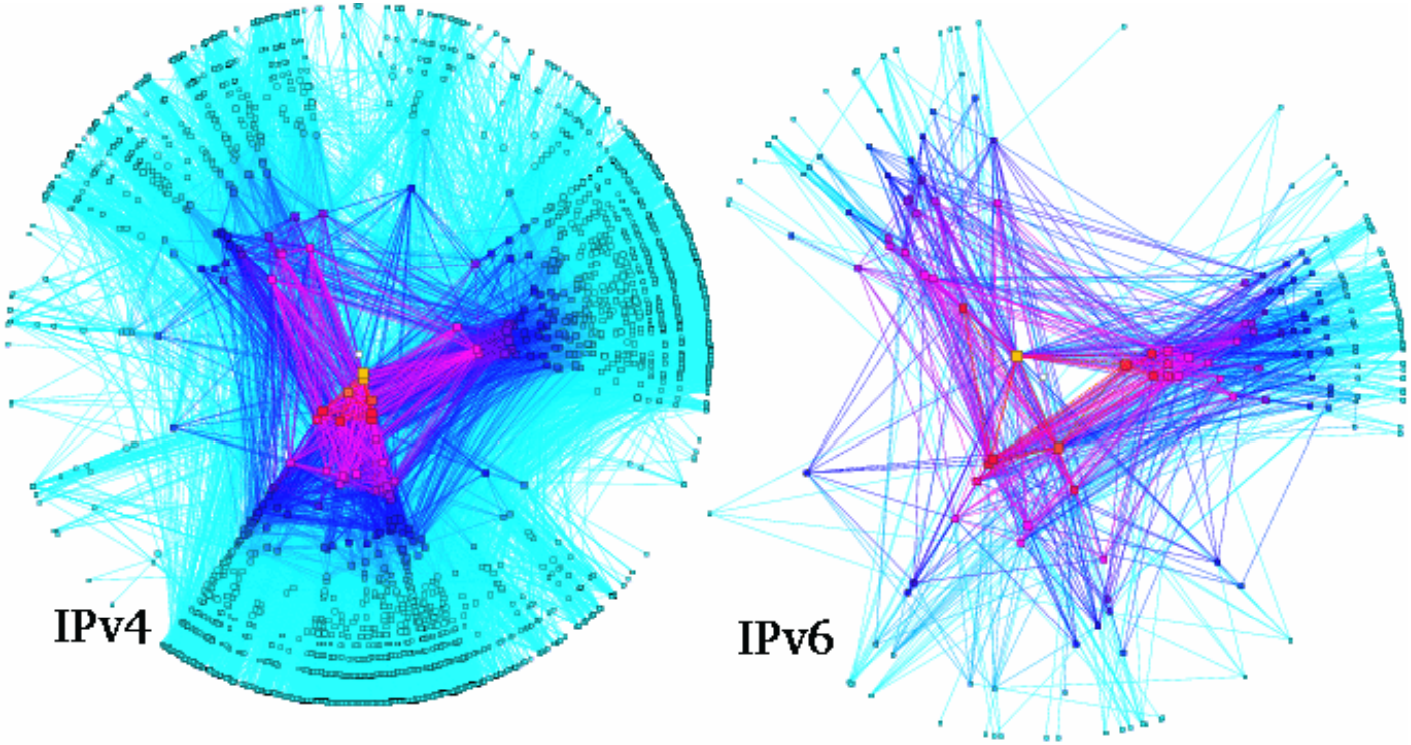
# 521261A

# Computer Networks I (spring 2008)

## Problem set #4



# Problem solving session #4

- Time: Thu 10.4.2008 at 8:15-9:45
- Location: L6

# Intermediate exam #4

- Time: Fri 11.4.2008 at 12:15-13:45
- Location: L3
- Material: lecture #08
- Remember to bring a photo ID and a calculator complying with the department's rules

# Problem #1 (Q)

(a) Suppose a process in host C has a UDP socket with port number 787. Suppose host A and host B each send a UDP segment to host C with destination port number 787. Will both of these segments be directed to the same socket at host C? If so, how will the process at host C know that these segments originated from two different hosts?

(b) Suppose that a web server runs in host C on port 80. Suppose this web server uses persistent connections and is currently receiving requests from two different hosts A and B. Are all these requests through the same socket at host C? If they are being passed through different sockets, do both sockets have port 80?

(c) Consider incoming TCP and/or UDP segments arriving at a server. Exactly N different destination port numbers are being used in the segments. The server acts only as a server, meaning that in the socket sense it does not initiate communication with any other computers as a client, but it only responds to incoming segments. Explain how many sockets are in use at the server.

# Problem #1 (A)

(a) Yes, both segments will be directed to the same socket. For each received segment, at the socket interface, the oeprating system will provide the process with the IP address of the host that sent the segment. The process can use the supplied IP addresses to determine the origins of the individual segments.

(b) For each persistent connection, the web server creates a separate "connection socket". Each connection socket is identified with a four-tuple (source IP address, source port number, destination IP address, destination port number). When host C receives an IP datagram, it examines these four fields in the datagram/segment to determine to which socket it should pass the payload of the segment. Thus, the requests from A and B pass through different sockets. The identifier for both of these sockets has 80 for the destination port number. However, the identifiers for these sockets have different values for the source IP addresses. Unlike UDP, when the transport layer passes a TCP segment's payload to the application process, it does not specify the source IP address, as this is implicitly specified by the socket identifier.

# Problem #1 (A)

(c) If arriving segments are all UDP segments, then the number of sockets in use at the server **equals the number of ports N**, because each arriving UDP segment will be received by the socket listening to the destination port number of the segment.

If arriving segments include also TCP segments, then the number of sockets in use at the server is going to be **larger than N**, because each welcoming socket and the socket created as a result of accepting a connection will have the same destination port number.

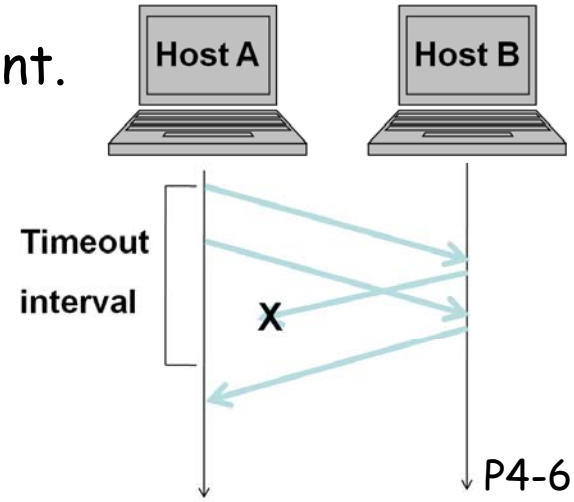
# Problem #2 (Q)

Host A and B are communicating over a TCP connection, and host B has already received from A all bytes up through byte 144. Suppose that host A then sends two segments to host B back-to-back, so that the first segment contains 20 bytes and the second segment contains 40 bytes. In the first segment the sequence number is 145, the source port number is 303 and the destination port number is 80. Host B sends an acknowledgment whenever it receives a segment from host A.

(a) If the first segment arrives before the second segment, in the acknowledgment of the first arriving segment, what are the acknowledgment number, the source port number, and the destination port number?

(b) If the second segment arrives before the first segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number?

(c) Suppose the two segments sent by A arrive in order at B. The first acknowledgment is lost and the second acknowledgment arrives after the first timeout interval, as shown in the figure right. Complete the diagram, showing all other segments and acknowledgments sent. Assume there is no additional packet loss. For each segment you add to the diagram, provide the sequence number and number of data bytes. For each acknowledgment that you add, provide the acknowledgment number.



# Problem #2 (A)

(a) Acknowledgment number is 165.

Source port number is 80.

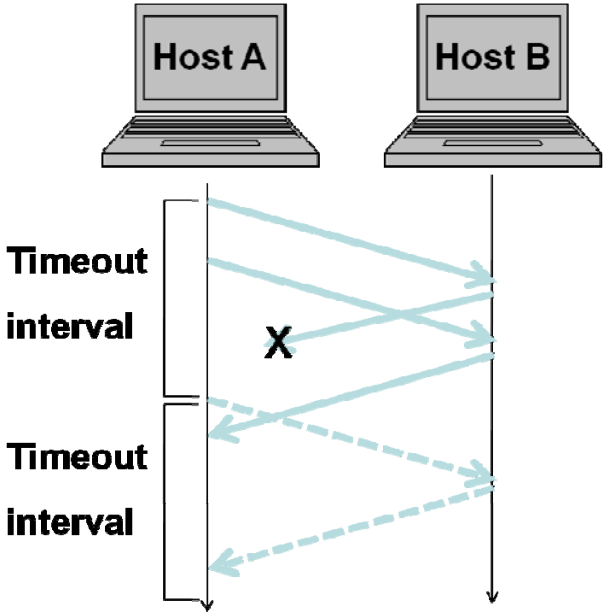
Destination port number is 303.

(b) The acknowledgment number is 145, indicating that the receiver is still waiting for bytes 145 and onward.

(c) Additional segments/acknowledgments are shown as dashed lines in the diagram right.

When the retransmission timer for the first segment expires, host A retransmits the first segment. The sequence number of this segment is 145 and it carries 20 bytes of data.

When the retransmitted segment arrives at host B, it sends an acknowledgment. The acknowledgment number is 205.



# Problem #3 (Q)

Given the plot of the size of the congestion window below:

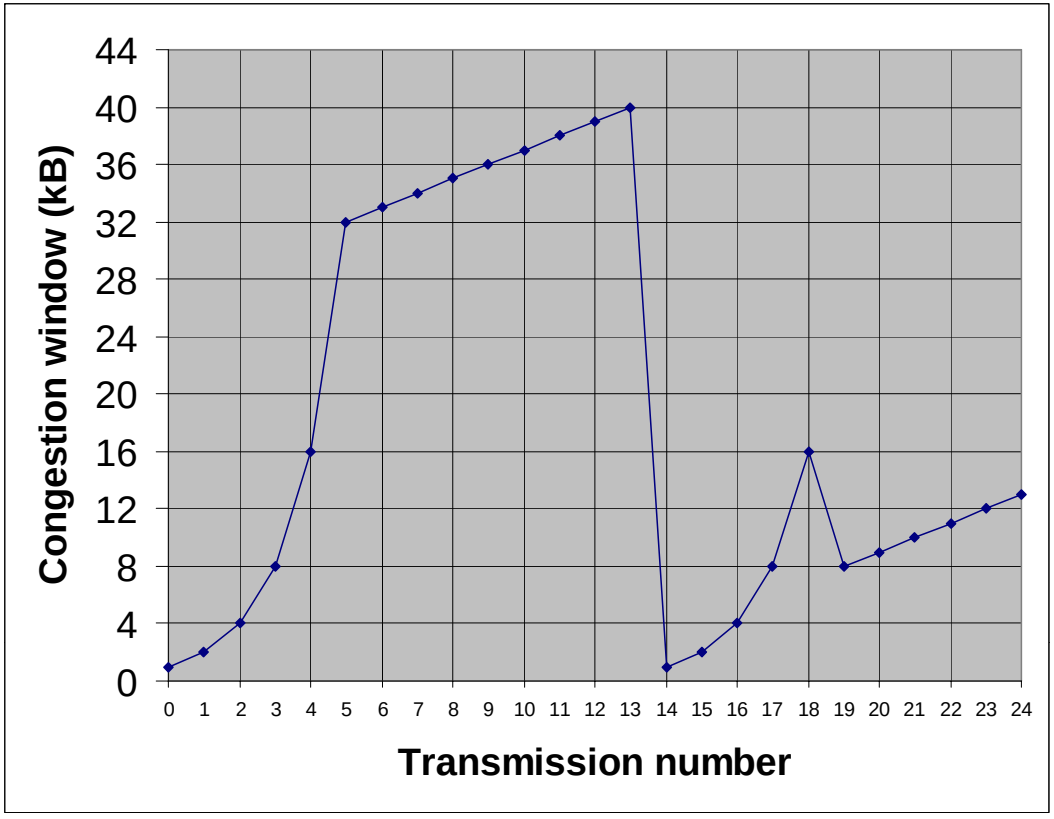
(a) what seems to be the maximum segment size?

(b) what happens to transmission #13?

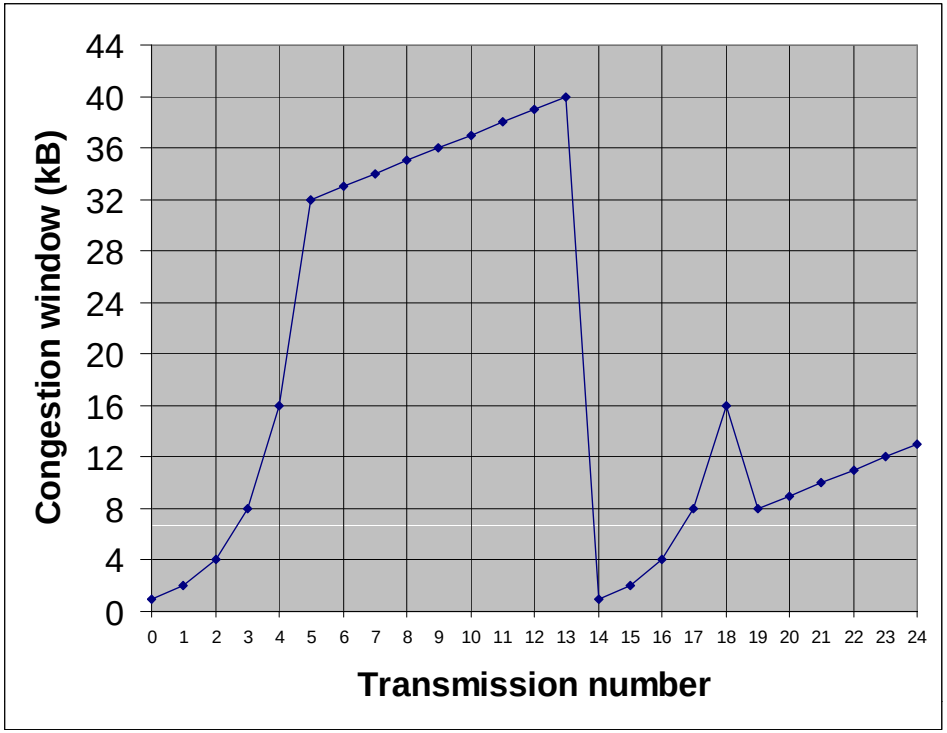
(c) what happens to transmission #18?

(d) what is the value of slow start threshold during transmission?

(e) explain the different phases of the transmission from #0 to #24.



# Problem #3 (A)




(a) 1 kB

(b) timeout

(c) triple duplicate ACKs

(d) 32 kB for transmissions #0 - #12,
20 kB for transmissions #13 - #17,
8 kb for transmissions #18-#24.

(e)

| | |
|---|---|
| 0-5 | CongWin is below Threshold (32 kB), sender in slow-start phase, window grows exponentially. |
| 6-12 | CongWin is above Threshold (32 kB), sender is in congestion-avoidance phase, window grows linearly. |
| 13 | Timeout occurs, Threshold is set to CongWin/2 (20 kB) and CongWin is set to 1 MSS |
| 14-17 | CongWin is below Threshold (20 kB), sender in slow-start phase, window grows exponentially. |
| 18 | triple duplicate ACK occurs, Threshold is set to CongWin/2 (8 kB) and CongWin is set to Threshold. |
| 19-24 | CongWin is above Threshold (8 kB), sender is in congestion-avoidance phase, window grows linearly. |

# Problem #4 (Q)

TCP connection has currently estimated RTT of 30 ms with a deviation of 3 ms. What is the value of the retransmission timer after:

(a) the next acknowledgement coming in after 28 ms?

(b) a timeout?

Use $\alpha$=0.875 in Jacobson's algorithm.

# Problem #4 (A)

**(a)**

**EstimatedRTT = $\alpha$xEstimatedRTT + (1-$\alpha$)xSampleRTT**
**= 0.875x30 ms + 0.125x28 ms = 29.75 ms**

**DevRTT = $\alpha$xDevRTT + (1-$\alpha$)*|SampleRTT-EstimatedRTT|**
**= 0.875x3 ms + 0.125x|28 ms - 30 ms| = 2.875 ms**

**RTO = EstimatedRTT + 4xDevRTT**
**= 29.75 ms + 4x2.875 ms = 41.25 ms**

**(b)**

Timeout $\rightarrow$ Karn's algorithm (double RTO)

**RTO = 2 * (30 + 4*3) = 84 ms**

# Problem #5 (Q)

(a) Consider sending an enormous file of L bytes from host A to host B over a TCP connection. Assume that the maximum segment size is 1460 bytes. What is the maximum value of L such that TCP sequence numbers are not exhausted (wrapped around)?

(b) What is the maximum line speed (in Mbps) at which a host can blast out 1300 byte TCP payloads without having the sequence numbers wrap around, when the maximum segment lifetime is 120 s? Take TCP, IP and Ethernet headers into account, and assume that Ethernet frames may be sent continuously.

# Problem #5 (A)

(a) Sequence number is 32 bit number, hence there are $2^{32}$ possible sequence numbers.

The sequence number is incremented by the number of bytes of data sent, i.e. MSS is irrelevant.

The maximum size file that can be sent from A to B is simply the number of bytes representable by **$2^{32}$ ~ 4.19 GB**.

# Problem #5 (A)

(b) The goal is to send $2^{32}$ payload bytes in 120 s, which equals ($2^{32}$ bytes / 1300 bytes) packets/120 s.

The TCP overhead is 20 bytes, the IP overhead is 20 bytes, and the Ethernet overhead is 26 bytes. This means that for 1300 bytes of payload, 1366 bytes must be sent.

Resulting line speed:

($2^{32}$ bytes / (120 s * 1300 bytes)) * 1366 bytes * 8 bits/byte = **300.9 Mbps**

# Problem #6 (Q)

(a) Consider the effect of using slow start on a line with 10 ms round-trip time and no congestion. The receive window is 24 kB and the maximum segment size is 2 kB. How long does it take before the first full window can be sent?

(b) Suppose that the TCP congestion window is set to 18 kB and a timeout occurs. How big will the window be if the next four transmissions are all successful? Assume that the maximum segment size is 1 kB.

# Problem #6 (A)

(a) RTT=10 ms, MSS=2 kB, no congestion, RecWin=24 kB

| transmission # | time (ms) | CongWin (kB) |
|---|---|---|
| 1 | 0 | 2 |
| 2 | 10 | 4 |
| 3 | 20 | 8 |
| 4 | 30 | 16 |
| 5 | 40 | 24 (RecWin) |

Answer: 40 ms

# Problem #6 (A)

(b) CongWin = 18 kB, MSS = 1 kB

timeout → CongWin = MSS, slow start

| transmission # | CongWin (kB) |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 3 | 4 |
| 4 | 8 |

Answer: 8 kB

# Problem #7 (Q)

A TCP machine is sending full windows of 65535 bytes over a 1 Gbps channel that has 10 ms one-way delay.

(a) What is the maximum throughput achievable?

(b) What is the line efficiency?

(c) What should the window size be for 100% efficiency?

# Problem #7 (A)

(a) One window can be sent every 20 ms (2x10 ms), i.e. 50 windows can be sent every second.

This gives maximum data rate of 50 windows/s x 65535 bytes/window * 8 bits/byte = **26.214 Mbps**

(b) Line efficiency is 26.214 Mbps / 1000 Mbps = **2.6%.**

(c) Window size = 1000 Mbps x 20 ms = 2.5 MB ☺

The window size equals BDP (bandwidth x delay product) which "fills the pipe" for 100% efficiency

# Problem #8 (Q)

Recall the idealized model for the steady-state dynamics of TCP (slides 08:87-88). In the period of time from when the connection's window size varies from W/2 to W, only one packet is lost at the very end of the period.

(a) Derive the loss rate (ratio of the number of packets lost over the number of packets sent) of the connection.

(b) Using the result of (a) show that if a connection has loss rate L, then its average bandwidth is approximately given by

$$\frac{1.22 \cdot MSS}{RTT\sqrt{L}}$$

# Problem #8 (A)

(a) Loss rate is the ratio of the number of packets lost over the number of packets sent. In a cycle, 1 packet is lost.

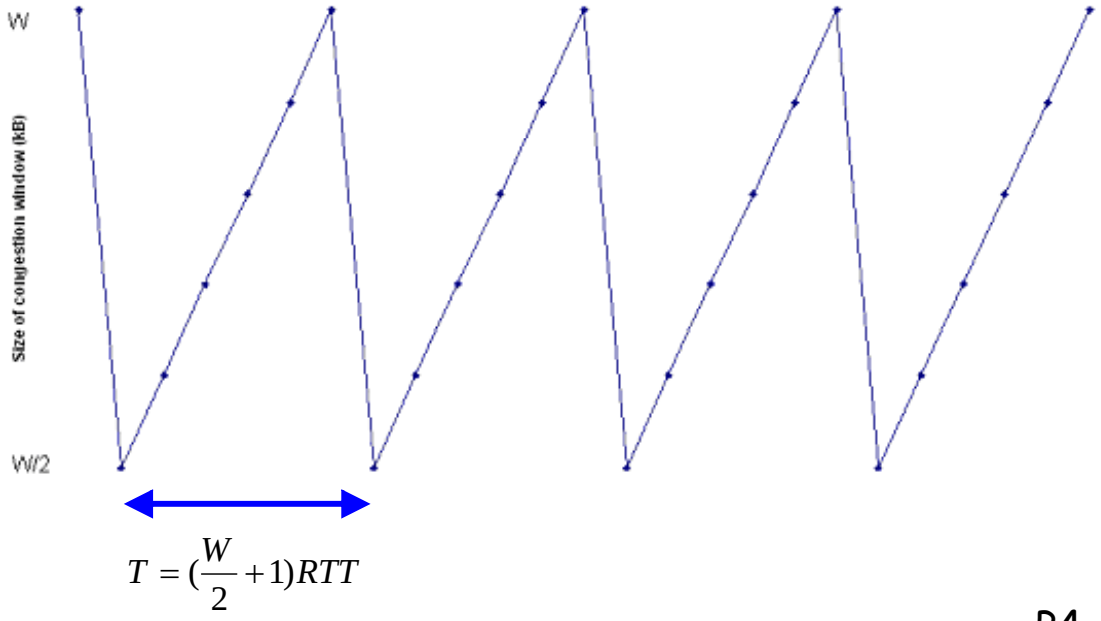
Number of packets sent:

$$N = \frac{W}{2} + \left(\frac{W}{2}+1\right) + \cdots + W = \sum_{n=0}^{W/2}\left(\frac{W}{2}+n\right)$$

$$= \left(\frac{W}{2}+1\right)\frac{W}{2} + \frac{W/2(W/2+1)}{2} = \left(\frac{W}{2}+1\right)\frac{3W}{4}$$

Loss rate = 1/N =

$$\frac{1}{\frac{3}{8}W^2 + \frac{3}{4}W}$$



# Problem #8 (A)

(b) Loss rate: $$L = \frac{1}{\frac{3}{8}W^2 + \frac{3}{4}W}$$

For W large $W^2 >> W$, thus $L \approx 8/(3W^2)$ or $W \approx \sqrt{(8/3L)}$

Average throughput (slide 08:90):

$$\frac{3W}{4RTT}MSS = \frac{3}{4}\sqrt{\frac{8}{3L}} \cdot \frac{MSS}{RTT} = \frac{1.22 \cdot MSS}{RTT \cdot \sqrt{L}}$$

# Problem #9 (Q)

A client requests and is delivered an object of size 100 kB from a server over a 10 Mbps link which has MSS of 536 bytes and RTT of 100 ms. Suppose that the transport protocol uses static windows with window size W.

(a) Determine the minimum amount of time needed for requesting and delivering the object from the server to the client.

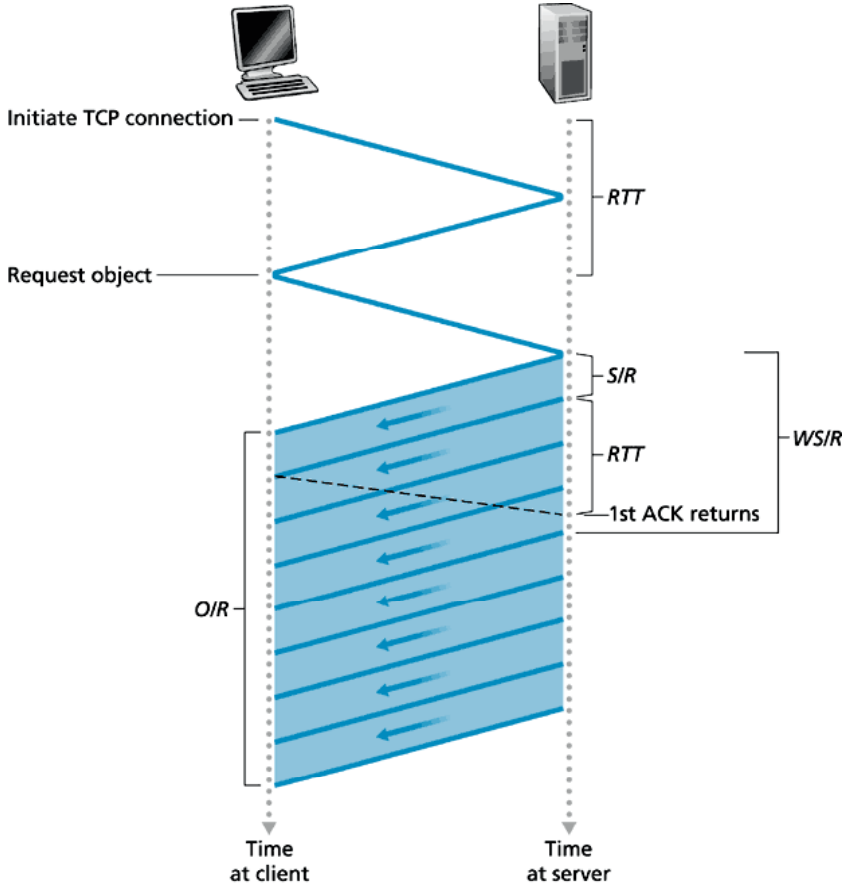
(b) Determine the minimum window size $W_{min}$, with which the time computed in (a) is achieved.

# Problem #9 (A)

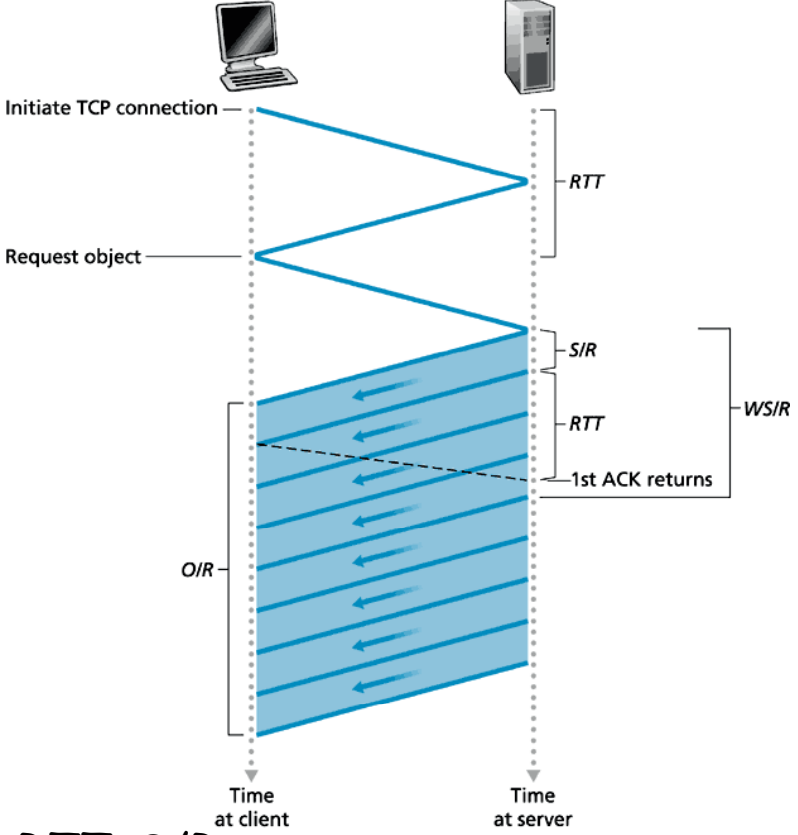
(a) The minimum amount of time needed corresponds to following three steps:

1. Client initiates TCP connection (1 RTT)
2. Client request for the object (request is piggybacked onto the third segment in the three-way handshake, 1 RTT),
3. The client begins to receive data from the server. It takes O/R period of time for the server to transmit the entire object, where O is the size of the object and R is the transmission rate

Thus, the minimum latency is
= 2RTT+O/R
= 2x100 ms + 100x8x$10^3$ b / 10x$10^6$ b/s
= 200 ms + 80 ms = **280 ms**



# Problem #9 (A)




(b) Let S denote the maximum segment size of 536 bytes. To achieve the minimum latency the server needs to receive an acknowledgement for the first segment in the first window before the server completes the transmission of the first window (otherwise server would stall while waiting for the acknowledgement and the minimum latency would not be achieved):

time to receive acknowledgement for the first segment: RTT+S/R

time to complete the transmission of the first window: $W_{min}S/R$

$RTT+S/R < W_{min}S/R$

$W_{min} > (RTT + S/R) / (S/R)$

$= RTT/(S/R) + 1$

$= 100\ ms / (536x8\ b/10x10^6\ b/s) + 1 = 233.2 + 1 = 234.2 \rightarrow$ **235**

Minimum window size in bytes = 235x536 = 125 960 bytes

# Problem #10 (Q)

Consider requesting and transmitting a 20000-byte object over a 10 Mbps link where RTT is 10 ms. Maximum segment size is 536 bytes. Take TCP connection establishment into account in your calculations. You can assume error-free transmission.

(a) Compute the minimum latency ignoring TCP slow start, i.e. TCP can blast away at full speed right after the connection is established.

(b) Compute the latency taking TCP slow start into account.

# Problem #10 (A)

(a) O=20000 bytes, R=10 Mbps, RTT=10 ms, S=536 bytes

minimum latency =
latency for connection setup (RTT) +
latency for request (RTT) +
latency for transmitting object (O/R)
= 2RTT + O/R
= 2 x 10 ms + 20000x8 bits / $10 \times 10^6$ bits/s
= 20 ms + 16 ms = 36 ms

# Problem #10 (A)

(b) number of segments = 20000 / 536 = 38 segments

| transmission number (k) | window size in segments ($2^{k-1}$) | total number of segments | idle time (ms) after transmission of this window |
|---|---|---|---|
| 1 | 1 | 1 | 10.00 (RTT) |
| 2 | 2 | 3 | 9.57 |
| 3 | 4 | 7 | 8.71 |
| 4 | 8 | 15 | 7.00 |
| 5 | 16 | 31 | 3.57 |
| 6 | 32 | 63 | 0 |

total idle time = 38.9 ms

$\rightarrow$ K=6 windows needs to be sent

# Problem #10 (A)

After transmitting a windows worth of data the server may stall (idle) while waiting an acknowledgement for the first segment of the window.

Let's compute minimum window size of W segments that the server does not manage to transmit fully before the ACK for the first segment arrives (i.e. the server does not have to idle before transmitting the next window):

$WS/R > RTT + S/R$

$W > RTT / (S/R) + 1 = 10 \cdot 10^6$ bits/s x 0.01s / (536x8 bits) + 1 = 23 + 1

$\rightarrow$ server will idle after first five transmissions in this problem

Let's compute the idle time for a window of $2^{k-1}$ segments, which is the difference of:
the time for the server to receive an acknowledgment for the first segment in the window (S/R+RTT)
and
the transmission time of the window ($2^{k-1}S/R$): $S/R + RTT - 2^{k-1}S/R$

The server idles after the transmissions of each of the first K-1 windows (server is done after transmitting the Kth window), hence the total idle time is $\sum_{k=1}^{K-1}\left[\frac{S}{R} + RTT - 2^{k-1}\frac{S}{R}\right]^+$

Latency = minimum latency + total idle time = $2RTT + O/R + \sum_{k=1}^{5}\left[\frac{S}{R} + RTT - 2^{k-1}\frac{S}{R}\right]^+$

= 36 ms + 38.9 ms = **74.9 ms**