

CS433/533: Computer Networks

Midterm 1

2/25/2002

6:30-8:30pm

Name:

Problem 1:

Problem 2:

Problem 3:

Problem 4:

Total:

1. Provide *short* answers to the following questions.

a) [5 points] Give two reasons why the Internet uses datagram switching instead of circuit switch.

Statistic multiplexing for bursty traffic
No setup latency, no state, simple
Robustness in the face of failure

b) [5 points] Caching is a commonly used technique to reduce traffic and latency. However, one potential drawback of caching is that cached data may be staled. What feature of HTTP allows a browser to guarantee the freshness of a web page when it uses its browser cache?

Conditional GET
Last-Modified
If-Modified-Since

d) [5 points] What are the functions of port numbers in the TCP header? In particular, why does TCP header include both source port number and destination port number?

Multiplexing/demultiplexing.
Since a server may listen at a port number to serve multiple clients, we also need source port number.

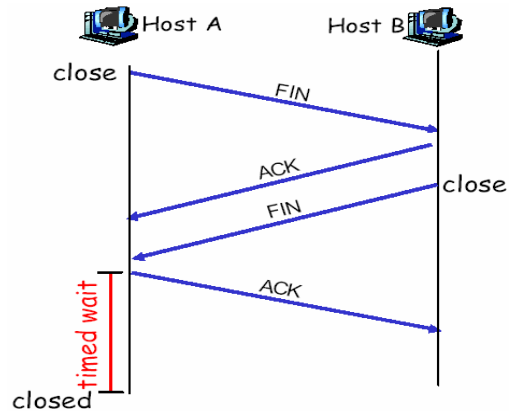
e) [5 points] To avoid any ambiguity, for a sliding window protocol with w bits as sequence number, we have to limit the window size. Since TCP sequence number is 32 bits, what is the maximum sliding window size for TCP?

Even though TCP acknowledgement is more like Go-back- n , its receiving window is more like selective repeat because the receiver will accept out-of-order segments. Thus you should use $M/2$, where $M = 2^{32}$. Even further, since the sliding window will also be limited by receiver flow control, which is 16 bits, the correct answer should be $2^{16}-1$ (because the maximum window size the receiver can advertise is $2^{16}-1$).

e) [5 points] e) [5 points] TCP FIN is considered to occupy one byte in the sequence number space. For example, after receiving a TCP FIN segment $\langle \text{FIN}, \text{seq}=x \rangle$, the receiver will send $\langle \text{ack}=x+1 \rangle$. One possibility is that FIN does not occupy any sequence number. Are there any issues of this approach?

Cannot distinguish between the ACKs for the last byte or FIN.

f) [5 points] For TCP, the side that makes the active close will go through the **time_wait** state, which lasts for two times the maximal time an IP packet can be stored in the Internet. The typical durations of a port in time_wait state are 60 seconds or 4 minutes. Give at least one reason for having the **time_wait** state.



Retransmit the last ACK if it is lost.

Avoid starting a new connection (this is why you see the Address in use error). Because if you start a new connection immediately and the last FIN is duplicated, this duplicated FIN may terminate the new connection.

g) [5 points] What does it mean when we say that window-based congestion control protocols are self-clocking?

The sending rate is limited by the rate ACKs return to the sender, which is the bottleneck rate.

When queue builds up, the rate of ACK is slowed down, and the data rate is automatically slowed down.

When we say self-clocking, we don't refer to the window adjustment scheme.

h) [5 points] TCP calculates its timeout as $SRTT + 4 \text{ Dev}$, where SRTT is the average value of measured round-trip time, and Dev is the deviation of the measured round-trip time. Describe the reasons for including Dev in calculating timeout value.

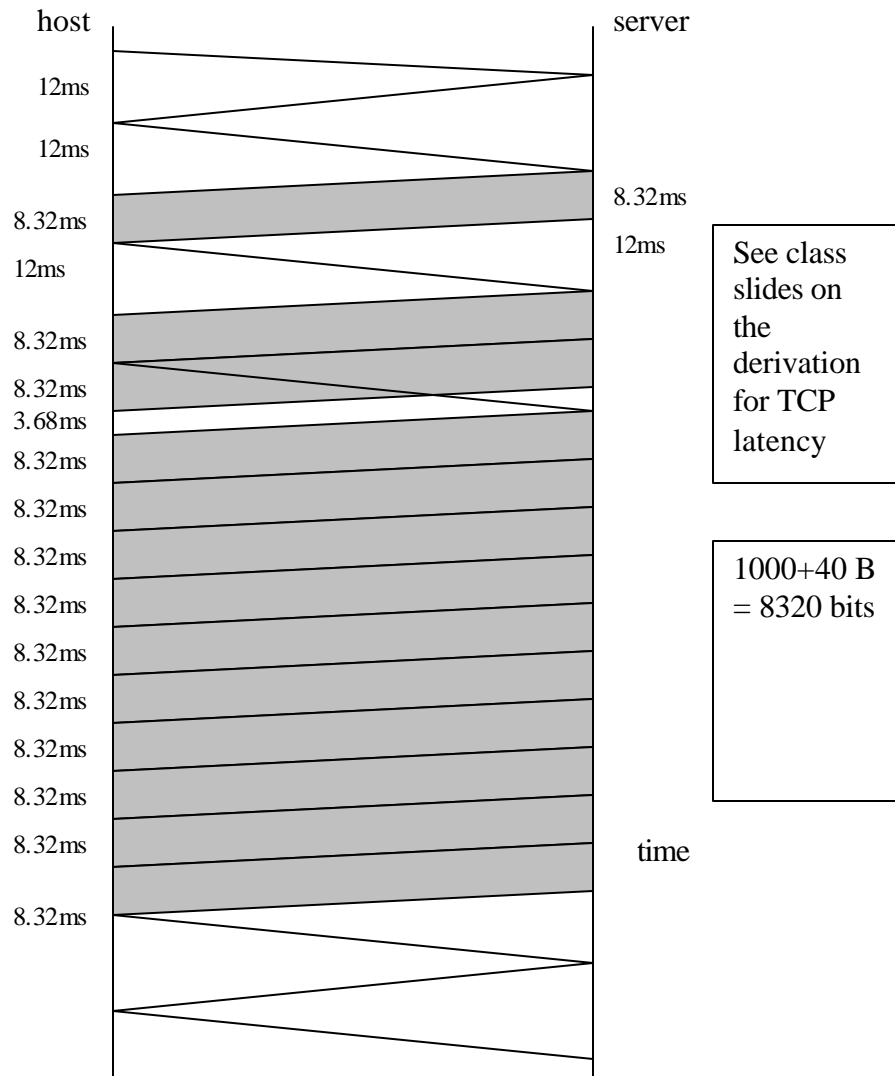
Since round-trip time fluctuates due to random queueing delay (because of cross traffic), in order to use a reasonable value for timeout but still have a low probability of timeout, we need to use Dev. Dev adapts to the fluctuations of the round-trip time.

2. Consider a user who accesses the Internet using a connection with a link speed of 1Mbps.

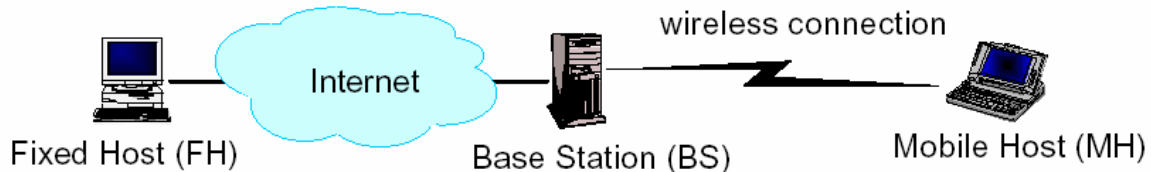
a) [5 points] Assume this is the first time that the user visits the home page of the CS department at Yale. The user types in the URL `http://www.cs.yale.edu/index.html` to its browser. Give the types of application packets you may capture for this transaction.

HTTP, DNS

b) [10 points] Let's only consider the latency of the TCP connection to download the web page. Assume the round-trip time between the host and the web server is 12ms; the size of the web page is 13K bytes; MSS of TCP is 1K bytes; TCP/IP header 40 bytes. For simplicity, you can ignore the transmission time of all short packets (e.g. ACK, HTTP request). For ease of calculation, you can assume $1K = 1000$, and $1M = 1,000,000$. Draw the space-time diagram for the TCP transaction, assuming no packet loss and no delayed ACK. Please label the delays on the diagram.



3. TCP congestion control does not work very well when a link has a high random packet loss rate. This is especially true in a wireless network environment. Consider below a reliable data transaction, which has a wireless link with a data packet loss rate of 1% (since control packets are smaller, assume their loss rate is negligible). Assume the MSS of TCP is 1K bytes and the TCP source has infinite data to send. Assume the round-trip time between the fixed host (FH) and the base station (BS) is 100ms, while the round-trip time between the BS and the mobile host (MH) is also 100ms.



a) [5 points] Assume we run a normal TCP between the FH and the MH, ignoring slow-start and timeout, using the approximate relationship $throughput \approx \frac{1.4S}{RTT\sqrt{p}}$, what is the throughput of the TCP connection?

Plug in the parameters of $S = 1\text{K bytes}$, $RTT = 200\text{ms}$, $p = 0.01$.

$$throughput \approx \frac{1.4 \times 8 \text{ Kbyte}}{0.2 \sqrt{0.01}} = 70 \text{ Kbytes/sec}$$

b) [5 points] One approach to improve the performance of the TCP connection is to use split-connection. In split-connection, one normal TCP connection is run between the FH and the BS and another connection is run between the BS and the MH. When the BS has received and acknowledged an in-order segment from the TCP connection between FH and BS, it starts to send the segment to the TCP connection between the BS and the MH. If we assume there is no packet loss between the FH and the BS, the throughput will be limited by the TCP connection between the BS and the MH. Similar to a), what is the throughput of split-connection? Does split-connection improve throughput?

Plug in the parameters of $S = 1\text{K bytes}$, $RTT = 100\text{ms}$, $p = 0.01$.

$$throughput \approx \frac{1.4 \times 8 \text{ Kbyte}}{0.1 \sqrt{0.01}} = 140 \text{ Kbytes/sec}$$

b) [5 points] What arguments will you use to argue against split-connection?

Change TCP semantics. If the BS crashes, the receiver may not be able to recover the segments acknowledged by the BS to the FH. Add delay and complexity to BS. This is a typical case of the end-to-end arguments.

4. In class, we have discussed the class of synchronous algorithms that adjust the sending rate $x_i(t)$ of flow i at time t :

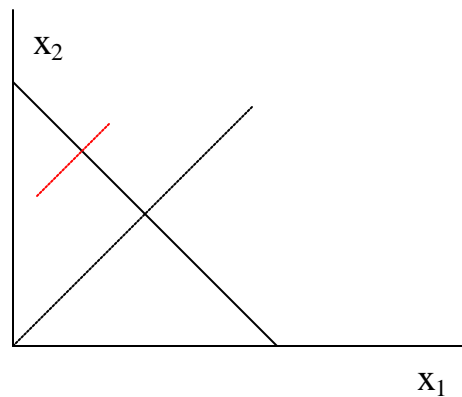
$$x_i(t+1) = \begin{cases} a_I + b_I x_i(t) & \text{if } \sum_i x_i(t) < C \\ a_D + b_D x_i(t) & \text{else} \end{cases}$$

where C is the capacity of the link.

a) [4 points] If we have $b_I = b_D = 1$, the algorithm is called Additive Increase Additive Decrease (AIAD). Under what conditions for a_I and a_D will the system be stable? Here by stable, we mean that when the total traffic exceeds link capacity, the flows will reduce their rates; when the total traffic is less than link capacity, the flows will increase their rates.

$$a_I > 0 \text{ and } a_D < 0$$

b) [6 points] Can AIAD achieve fairness? Here by fairness, we mean that when two flows start with different initial sending rates, the difference will eventually become zero. Please use the state space diagram to justify your answer.

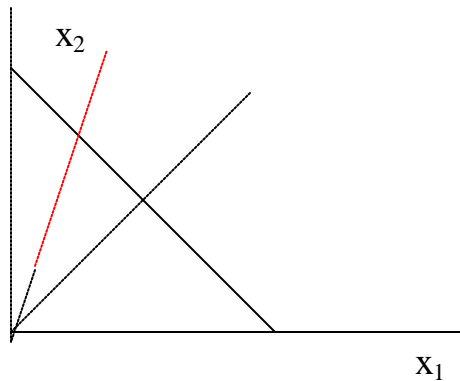


No. The rates will fluctuate along the red line.

c) [4 points] If we have $a_I = a_D = 0$, the algorithm is called Multiplicative Increase Multiplicative Decrease (MIMD). Under what conditions for b_I and b_D will the system be stable?

$$b_I > 1 \text{ and } 0 < b_D < 1$$

d) [6 points] Can MIMD achieve fairness?



No. The rates will fluctuate along the line.

e) [5 points] If we have $a_I = 0$, $b_I > 1$, $a_D < 0$, $b_D = 1$, the algorithm is called Multiplicative Increase Additive Decrease (MIAD). Can MIAD achieve fairness?

No. MIAD will actually move away.

