Study Guide Problems (Mid-Term #1)

# Chapter 1

**R11.** Suppose there is exactly one packet switch between a sending host and a receiving host. The transmission rates between the sending host and the switch and between the switch and the receiving host are *R1* and *R2*, respectively. Assuming that the switch uses store-andforward packet switching, what is the total end-to-end delay to send a packet of length *L*? (Ignore queuing, propagation delay, and processing delay.)

It takes L/R seconds to transmit L bits of data over a link of bandwidth R bits per second. Hence, the total delay is L/R1 +L/R2.

**R16.** Consider sending a packet from a source host to a destination host over a fixed route. List the delay components in the end-to-end delay. Which of these delays are constant and which are variable?

Propagation delay constant, transmission delay constant, queuing delay variable

**R18.** How long does it take a packet of length 1,000 bytes to propagate over a link of distance 2,500 km, propagation speed 2.5\*108m/s, and transmission rate 2 Mbps? More generally, how long does it take a packet of length *L* to propagate over a link of distance *d*, propagation speed *s*, and transmission rate *R* bps? Does this delay depend on packet length? Does this delay depend on transmission rate?

2500km = 2500000m

Propagation speed = 2500000/ 2.5\*10^8 = 0.01s

Transmission rate = 2\*10^6bps

8000/(2\*10^6) = 4\*10^-3s

Total = 4\*10^-3s + 0.01s

**P6.** This elementary problem begins to explore propagation delay and transmission delay, two central concepts in data networking. Consider two hosts, A and B, connected by a single link of rate *R* bps. Suppose that the two hosts are separated by *m* meters, and suppose the propagation speed along the link is *s* meters/sec. Host A is to send a packet of size *L* bits to Host B.

1. Express the propagation delay, *dprop*, in terms of *m* and *s*.

m/s

1. Determine the transmission time of the packet, *dtrans*, in terms of *L* and *R*.

L/R

1. Ignoring processing and queuing delays, obtain an expression for the end-to-end delay.

m/s+L/R

1. Suppose Host A begins to transmit the packet at time t=0. At time *t= dtrans*, where is the last bit of the packet?

The last bit just leaved the Host A and just enter the link

1. Suppose *dprop* is greater than *dtrans*. At time t= *dtrans*, where is the first bit of the packet?

The first bit is still on the link between A and B

1. Suppose *dprop* is less than *dtrans*. At time t= *dtrans*, where is the first bit of the packet?

The first bit already arrive the other side of the link

1. Suppose s=2.5\*108, L=120bits, and R=56kbps. Find the distance *m* so that *dprop* equals *dtrans.*

propagation speed : 2.5\*108meters/sec

Packet size: 120bits

Rate: 56kbps = 56000bps

M:x

x/2.5\*10^8 = 120/56000 = 535714m

**P24.** Suppose you would like to urgently deliver 40 terabytes data from Boston to Los Angeles. You have available a 100 Mbps dedicated link for data transfer. Would you prefer to transmit the data via this link or instead use FedEx over-night delivery? Explain.

40Tbytes = **320,000,000,000,000 bp**

100mpbs = 1\*10^8 pbs

**320,000,000,000,000 bp** /1\*10^8 pbs = 3200000s = 37 days

So fedex

# Chapter 2

**R13**. Describe how Web caching can reduce the delay in receiving a requested object. Will Web caching reduce the delay for all objects requested by a user or for only some of the objects? Why?

Web caching reduces overall delays due to two reasons. First, caches are closer to users than the originating servers (origin servers) of content, so serving content from caches incurs smaller propagation delays. A more subtle reason is that it is more likely that there is a higher bandwidth path available between the cache and the user (compared to between the origin server and the user), hence the transmission delays to obtain an object from the cache are lower than those of the origin server.

Web caching is not useful for dynamic or personalized objects that may not be stored (by the cache) independently of the origin server. An example of personalized content is a bank balance, which is not shared across users, and hence there is little benefit to caching.

**R14.** Telnet into a Web server and send a multiline request message. Include in the request message the *If-modified-since:* header line to force a response message with the *304* *Not Modified* status code.

 telnet <domain name of web server> <port number>

**P1.** True or false?

1. A user requests a Web page that consists of some text and three images. For this page, the client will send one request message and receive four response messages.

False. An application must send an individual HTTP-level request for distinct objects, since the protocol format only allows one resource to be requested per HTTP message. (Note that this restriction does not contradict with putting multiple HTTP-level requests into a single persistent TCP connection or even a single packet.)

1. Two distinct Web pages (for example, www.mit.edu/research.html and www.mit.edu/students.html ) can be sent over the same persistent connection.

T

1. With nonpersistent connections between browser and origin server, it is possible for a single TCP segment to carry two distinct HTTP request messages.

False. With non-persistent connections, two HTTP request messages cannot even be on the same TCP connection, let alone the same TCP segment.

1. The *Date:* header in the HTTP response message indicates when the object in the response was last modified.

False. The Date: header corresponds to the time the HTTP server obtained the resource from its local filesystem or storage.

1. HTTP response messages never have an empty message body.

False For example, if the response is for a cached object that was unmodified, the message (entity) body is empty.

**P4.** Consider the following string of ASCII characters that were captured by Wireshark when the browser sent an HTTP GET message (i.e., this is the actual content of an HTTP GET message). The characters *<cr><lf>* are carriage return and line-feed characters (that is, the italized character string *<cr>* in the text below represents the single carriage-return character that was contained at that point in the HTTP header). Answer the following questions, indicating where in the HTTP GET message below you find the answer. *GET /cs453/index.html HTTP/1.1<cr><lf>Host: gai*

*a.cs.umass.edu<cr><lf>User-Agent: Mozilla/5.0 (*

*Windows;U; Windows NT 5.1; en-US; rv:1.7.2) Gec*

*ko/20040804 Netscape/7.2 (ax) <cr><lf>Accept:ex*

*t/xml, application/xml, application/xhtml+xml, text*

*/html;q=0.9, text/plain;q=0.8, image/png,\*/\*;q=0.5*

*<cr><lf>Accept-Language: en-us, en;q=0.5<cr><lf>Accept- Encoding: zip, deflate<cr><lf>Accept-Charset: ISO*

*-8859-1, utf-8;q=0.7,\*;q=0.7<cr><lf>Keep-Alive: 300<cr> <lf>Connection:keep-alive<cr><lf><cr><lf>*

1. What is the URL of the document requested by the browser?

gaia.cs.umass.edu/cs453/index.html

1. What version of HTTP is the browser running?

*HTTP/1.1*

1. Does the browser request a non-persistent or a persistent connection?

persistent

1. What is the IP address of the host on which the browser is running?

those headers are not shown here.

1. What type of browser initiates this message? Why is the browser type needed in an HTTP request message?

The initiating browser is *Mozilla/5.0* The browser type may be needed for the web server to provide HTML and other scripts which may be browser-specific. While the HTTP protocol is standardized, how browsers interpret the contents of an HTTP response is not standardized, hence web servers may need to specialize the response

**P5.** The text below shows the reply sent from the server in response to the HTTP GET message in the question above. Answer the following questions, indicating where in the message below you find the answer.

*HTTP/1.1 200 OK<cr><lf>Date: Tue, 07 Mar 2008*

*12:39:45GMT<cr><lf>Server: Apache/2.0.52 (Fedora)*

*<cr><lf>Last-Modified: Sat, 10 Dec2005 18:27:46*

*GMT<cr><lf>ETag: ”526c3-f22-a88a4c80”<cr><lf>Accept- Ranges: bytes<cr><lf>Content-Length: 3874<cr><lf> Keep-Alive: timeout=max=100<cr><lf>Connection:*

*Keep-Alive<cr><lf>Content-Type: text/html; charset= ISO-8859-1<cr><lf><cr><lf><!doctype html public ”-*

*//w3c//dtd html 4.0 transitional//en”><lf><html><lf>*

*<head><lf> <meta http-equiv=”Content-Type” content=”text/html; charset=iso-8859-1”><lf> <meta*

*name=”GENERATOR” content=”Mozilla/4.79 [en] (Windows NT*

*5.0; U) Netscape]”><lf> <title>CMPSCI 453 / 591 /*

*NTU-ST550ASpring 2005 homepage</title><lf></head><lf>*

*<much more document text following here (not shown)>*

1. Was the server able to successfully find the document or not? What time was the document reply provided?

Yes, *Tue, 07 Mar 2008 12:39:45GMT*

1. When was the document last modified?

*Sat, 10 Dec2005 18:27:46 GMT*

1. How many bytes are there in the document being returned?

*3874*

1. What are the first 5 bytes of the document being returned? Did the server agree to a persistent connection?

The actual document is returned in the entity body, which is always preceded by two empty lines (you’ll see two carriage-return line-feed characters <cr><lf><cr><lf>). The first bytes of the document begin with <!doctype...

The server indeed agreed to a persistent connection,

**P7.** Suppose within your Web browser you click on a link to obtain a Web page. The IP address for the associated URL is not cached in your local host, so a DNS lookup is necessary to obtain the IP address. Suppose that *n* DNS servers are visited before your host receives the IP address from DNS; the successive visits incur an RTT of RTT1, …, RTTn. Further suppose that the Web page associated with the link contains exactly one object, consisting of a small amount of HTML text. Let RTT0 denote the RTT between the local host and the server containing the object. Assuming zero transmission time of the object, how much time elapses from when the client clicks on the link until the client receives the object?

Initiating a connection incurs RTT0, and fetching the object takes another RTT0. Hence, the total time is 2RTT0 +RTT1 +···+RTTn

**P13.** What is the difference between *MAIL FROM* : in SMTP and *From* : in the mail message itself?

**Mail FROM in SMTP contains <name>@<sender-**domain>, and it is a command to an SMTP mail server. From in the mail message is a header which informs the user agent how to render an email message

**P20.** Suppose you can access the caches in the local DNS servers of your department. Can you propose a way to roughly determine the Web servers (outside your department) that are most popular among the users in your department? Explain.

There are a few approaches one could use. Many caches evict old items based on  
recency of use (i.e., evict the least recently used items (LRU)). Assuming the DNS cache is limited in space, the longevity of a domain in the cache indicates how recent and frequent accesses to this domain are. Another possibility is using the frequency with which the domain reappears in the cache after a possible eviction. Picking the highest longevity or the most frequent re-appearances are two methods which could provide an idea of the most popular domains accessed by the users in the department.

**P21.** Suppose that your department has a local DNS server for all computers in the department. You are an ordinary user (i.e., not a network/system administrator). Can you determine if an external Web site was likely accessed from a computer in your department a couple of seconds ago? Explain.

Check the website loading time

**P27.** Consider a DASH system for which there are *N* video versions (at *N* different rates and qualities) and *N* audio versions (at *N* different rates and qualities). Suppose we want to allow the player to choose at any time any of the *N* video versions and any of the *N* audio versions.

1. If we create files so that the audio is mixed in with the video, so server sends only one media stream at given time, how many files will the server need to store (each a different URL)?

N^2

1. If the server instead sends the audio and video streams separately and has the client synchronize the streams, how many files will the server need to store?

2n

# Chapter 9

**R2.** There are two types of redundancy in video. Describe them, and discuss how they can be exploited for efficient compression.

Spatial: send two values: color and number of repeated values (N)

avoid sending repeated data for each pixel.

Tempora: send only differences from frame i to i + 1

reduce the amount of information transmitted or stored by only recording and transmitting the differences between the frames.

**R3.** Suppose an analog audio signal is sampled 16,000 times per second, and each sample is quantized into one of 1024 levels. What would be the resulting bit rate of the PCM digital audio signal?

The bit rate of the signal is the number of samples multiplied by the number of bits per sample. Each sample is quantized into 1024 levels, which corresponds to log2(1024) = 10 bits. Hence, the bit rate of the resulting audio signal is 16000 \* 10 = 160000 bits/sec

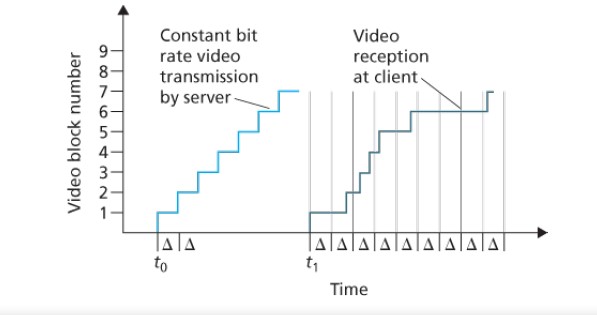
**R8.** Consider the simple model for HTTP streaming. Suppose the server sends bits at a constant rate of 2 Mbps and playback begins when 8 million bits have been received. What is the initial buffering delay *tp*?

2Mbps = 2\*10^6 bps

8 million bit = 8000000

Buffering delay = 8000000/(2\*10^6) = 4 sec

**P1.** Consider the figure below. Similar to our discussion of **Figure 9.1**, suppose that video is encoded at a fixed bit rate, and thus each video block contains video frames that are to be played out over the same fixed amount of time, Δ. The server transmits the first video block at *t0*, the second block at *t0 +* Δ *,* the third block at *t0 +* 2Δ*,* and so on. Once the client begins playout, each block should be played out Δ time units after the previous block.



1. Suppose that the client begins playout as soon as the first block arrives at *t1*. In the figure below, how many blocks of video (including the first block) will have arrived at the client in time for their playout? Explain how you arrived at your answer.

The kth block will have arrived in time if it arrives on or before time t1 +tp + (k −1) ∗∆,  
which is the time instant at which that block must be played out (due to the continuous playout constraint), where tp is the initial playout delay. In this part, tp = 0.

The picture shows 7 blocks overall. Assuming the first block arrived on time, only the 4th, 5th, and 6th blocks arrived on or before their intended time instant of playout.

1. Suppose that the client begins playout now at *t1 +* Δ. How many blocks of video (including the first block) will have arrived at the client in time for their playout? Explain how you arrived at your answer.

Due to the initial playout delay of ∆, the intended time instant of playout has shifted ∆to  
the right of what it was in part (a). With this new reference, all blocks except the 7th block have arrived before the intended time instant of playout.

1. In the same scenario at (b) above, what is the largest number of blocks that is ever stored in the client buffer, awaiting playout? Explain how you arrived at your answer.

The maximum buffer is two blocks, between times t1 + 3∆and t1 + 4∆. This can be iden-  
tified by drawing the “playout staircase”, i.e., the staircase corresponding to when the blocks are supposed to be played out after the initial playout delay. This is a staircase that begins its first vertical step at time t1 +∆and rises vertically one block for each interval of ∆. The largest observed gap between the staircase of video reception and the staircase of playout provides the maximum amount buffered in the client.

1. What is the smallest playout delay at the client, such that every video block has arrived in time for its playout? Explain how you arrived at your answer.

With the current playout delay (from part (b)), block 7 has the largest lag between the  
intended playout time and the video reception time. Assuming the playout delay can only be set in multiples of ∆, the minimum playout delay that can prevent block 7 from missing its playout time is 3∆.

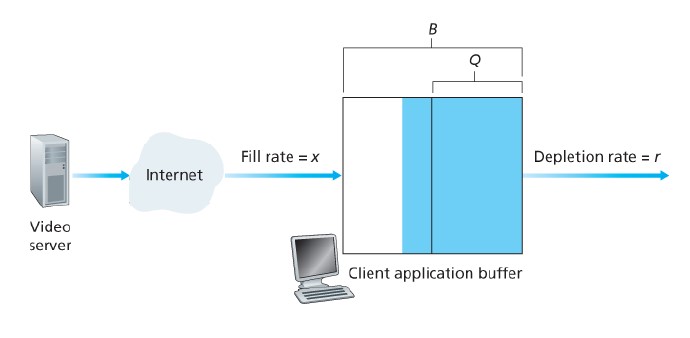
**P2.** Recall the simple model for HTTP streaming shown in **Figure 9.3** . Recall that *B* denotes the size of the client’s application buffer, and *Q* denotes the number of bits that must be buffered before the client application begins playout. Also *r* denotes the video consumption rate. Assume that the server sends bits at a constant rate *x* whenever the client buffer is not full.

1. Suppose that x<r. As discussed in the text, in this case playout will alternate between periods of continuous playout and periods of freezing. Determine the length of each continuous playout and freezing period as a function of *Q*, *r*, and *x*.

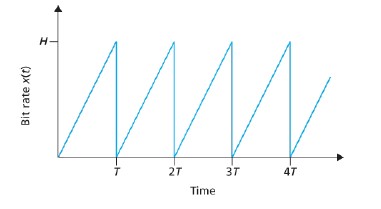
The buffer is draining at a rate of r-x. So the duration of playout is Q/(r-x). When freezing, the r will stop, and the period for x to fill up the Q is Q/x

1. Now suppose that x>r. At what time t=tf does the client application buffer become full?

When x > r, the buffer is filling at a rate of x−r after the initial playout delay. It takes time (B−Q)/(x−r) to fill the buffer after the initial playout delay tp.



**P3.** Recall the simple model for HTTP streaming shown in **Figure 9.3** . Suppose the buffer size is infinite but the server sends bits at variable rate *x*(*t*). Specifically, suppose *x*(*t*) has the following saw-tooth shape. The rate is initially zero at time t=0 and linearly climbs to *H* at time t=T. It then repeats this pattern again and again, as shown in the figure below.



1. What is the server’s average send rate?

Since the rate curve is linear, the average send rate is just the average of the two extremes, i.e., H/2. ((H\*T)/2)/T

1. Suppose that Q=0, so that the client starts playback as soon as it receives a video frame. What will happen?

The initial rate x(t ≈0) will not be large enough to support continuous playout, since for  
any video playout rate r there is a finite interval over which x <r. During this time, the video playout will freeze

1. Now suppose Q>0 and HT/2≥Q. Determine as a function of *Q*, *H*, and *T* the time at which playback first begins.

The fact that HT/2 ≥Q means that over a period of duration T, the amount of data transmit-  
ted into the video buffer H/2 ·T is larger than the video occupancy Q required for initial playout. Hence, playout can begin sometime within duration [0,T] itself. However, we need to compute as a function of t ∈[0,T] how much data is sent in aggregate by the sender with the sending rate pattern shown.

Since the slope of the rate curve is H/T, at time t, the rate is t ·H/T. The total amount of data transmitted during this time is just the area of a triangle with height t ·H/T and base t, i.e., Ht2/2T. Playout begins at time t when this quantity equals Q, i.e., t = √2QT/H.

Base: t, Height t\*H/T, area = t^2H/2T = Q

1. Suppose H>2r and Q=HT/2. Prove there will be no freezing after the initial playout delay.

HT/2 is the amount of video transmitted by the sender in one “see-saw” cycle of its rate  
variation over duration T. The fact that HT/2 = Q means that over the full first cycle, the sender will fill the buffer to occupancy Q. This is enough buffering to last the video client one full cycle of playout, since r <H/2. In this cycle of playout, the sender has already replenished the video buffer by HT/2 = Q. Hence, the video client will never stall after the initial playout delay.

1. Suppose H>2r. Find the smallest value of *Q* such that there will be no freezing after the initial playback delay.

r < H/2, that means the drain rate is smaller than our average bit rate, therefore if the video client can see through one cycle of the rate pattern without stalls after the initial playout,

1. Now suppose that the buffer size *B* is finite. Suppose H>2r. As a function of *Q*, *B*, *T*, and *H*, determine the time t=tf when the client application buffer first becomes full.