NAME:

Login name:

Computer Science 461
Midterm Exam
March 30, 2009
1:30-2:50pm

This test has six (6) questions. Put your name on ***every page***, and write out and sign the Honor Code pledge before turning in the test.

Please look through all of the questions at the beginning to help in pacing yourself for the exam. The exam has 100 points and lasts for 80 minutes, so the number of minutes spent per question should be *less* than its point value. You should spend no more than 10-12 minutes per question. Show your work for all problems. Partial credit will often be given.

"I pledge my honor that I have not violated the Honor Code during this examination."

Question	Score
1	/ 15
2	/ 16
3	/ 17
4	/ 14
5	/ 16
6	/ 22
Total	/ 100

QUESTION 1: Transporting a file (15 POINTS)

Suppose your boss calls you up while visiting a client and says he left a 250,000 byte file on a server at the office, and he needs you to send it to him. The connection between the office and his current site is 2 million bits/second.

1A. [2 points] Assuming that all bandwidth is consumed by your data (e.g., there aren't any packet headers, there's no other traffic on the link, etc.) and that you can immediately start sending data at the maximum rate, how long will it take to transmit the file? (Easy)

250,000 bytes * (8 bits / 1 byte) * 1/2,000,000 bits/s = 1 second

1B. [5 points] Now assume that you are sending the file via TCP, with a maximum segment size of 1000 bytes. You have to follow TCP's transport algorithm, so you start transmitting the file at a low rate. How many network round trip times (RTTs) will it take to transmit the entire file? Again assume that all bandwidth is consumed by your data.

TCP needs to first establish a connection via a handshake (SYN/SYN-ACK) - 1 RTT

TCP then enters slow-start to transmit the file, so we run slow-start either until we've transmitted the file or until we experience a loss (at 2Mbps). In slow-start, TCP will initially transmit a window of 1 MSS (=1000 bytes) per RTT, then double the window size each round (RTT). So, we have:

1 MSS, 2 MSS, 4, 8, 16, 32, 64, 128

The sum of the above 8 transfers is 255 MSS = 255,000 bytes > 250,000 byte file, so we declare ourselves finished after the 8th RTT.

Thus, the total time to transmit the file would be 1 + 8 = 9 RTT.

Note: One might debate what "transfer" means, so we also accepted 7.5 RTTs for the number of runs in slow-start, as the final transfer of 128 might not strictly need to ACK. We did not take off any points if people also included 1 RTT for FINs.

1C. [4 points] Let's now consider how TCP reacts when it encounters packet loss (the "basic" variant we learned in class). Your boss now wants you to transmit a much larger file (say, 1 GB). This time around, after TCP reaches an instantaneously transmission rate of 32,000 bytes per second, you notice that a single packet is lost. Assuming that you don't currently have any other packets in transmission, what is the instantaneous rate TCP will send at after noticing that loss? How long (in terms of RTTs) will it take to reach 32,000 bytes per second again?

This question was meant to seek how TCP reacts to detecting loss via a triple duplicate ACK. In that case, TCP reacts by "MD" -- multiplicative decrease -- cutting its window size by half and continuing by "AI" -- additive increase.

So the next transmission rate is 16,000 bytes per second, and it will take 16 RTTs to recover to its prior levels (each RTT increases by 1 MSS = 1000 bytes).

1D. [4 points] In the previous example of 1c, would anything be different if all packets were lost after you reached 32,000 bytes per second, instead of just a single packet? If so, what is the next instantaneous rate TCP will send at in this example, and how long (in terms of RTTs) would it now take to reach 32,000 bytes per second again?

Because all packets are lost, we only detect loss through a timeout. This causes the sender's window to be reduced to a single MSS (1000 bytes / RTT), at which time it enters "slow-start restart". Namely, it will perform slowstart until half of its prior cwnd (16,000 bytes/second), then do additive-increase from 16,000 to 32,000 bytes/second.

(In fact, the question as written didn't include quite enough information, as you actually do need to know the RTT in this example in order to calculate the window size for "16,000 bytes / seconds", i.e., if RTT is 500ms, window size is 8 MSS/RTT, while window size would be 16 MSS/RTT if RTT is 1s. Everybody just assumed implicitly that RTT=1s, and that the next rate after the timeout was 1000 bytes/sec. We initially had written these questions with an explicit RTT and asked you to solve the question in terms of time, rather than RTTs, but subsequently removed this information to reduce needed calculations.)

That said, assuming RTT=1s, then the answer is:

Slow-start from 1 MSS to 16 MSS = 4 RTTs, then additive increase 16-32 MSS = 16 RTTs

Total = 20 RTTs (we also accepted 21 RTTs given various ways to "count" these values)

QUESTION 2: Sockets (16 POINTS)

2A. [10 points] In the first assignment, you used the socket API to write a simple client/server. For this question, there is a single client communicating with a single server using TCP. The client sends a very small amount of data using the send() function, and the server receives the data with a recv() call. Each of the following scenarios shows a different order in which the socket calls are called on both the client and the server.

Assume all system calls are blocking, and there may be some latency between the client and server. We write the system call when it is initiated by the client and/or server; it returns sometime between then and when the next local system call is executed.

For each scenario, answer the following: Can the server ever get the data sent by the client? (Yes or No). If not, explain why not.

i)

Client	Server
	socket()
socket()	
	bind()
connect()	
	listen()
send()	
	accept()
close()	
	recv()
	close()

No. connect() wouldn't successfully return, allowing you to send(), before client accepts().

ii)

Client	Server
socket()	
connect()	
send()	
	socket()
	bind()
	listen()
	accept()
	recv()
close()	
	close()

No. Same explanation as (i). Also valid reason is that server isn't listening() when client attempts to connect().

iii)

Client	Server
	socket()
socket()	
	bind()
	listen()
	accept()
connect()	
send()	
	recv()
close()	
	close()

Yes.

iv)

Client	Server
socket()	
	socket()
	bind()
	listen()
	accept()
	recv()
connect()	
send()	
close()	
	close()

No. Accept() wouldn't return with a valid file descriptor until client has connected().

v)

Client	Server
socket()	
	socket()
connect()	
	bind()
	listen()
	accept()
	recv()
send()	
close()	
	close()

Yes.

2B. [3 points] Suppose application A is using a stream (TCP) socket to transfer data to application B on a remote host. Suppose application A writes (sends()) data into the socket buffer ten times.

i) Can the underlying network stack transmit *more* than ten data packets? If so, why?

Yes. If a write is larger than the MSS, then TCP might transmit it within two packets. Certainly if your total number of written data is larger than 10 MSSs, you are guarnateed to have more than 10 TCP packets. (Also accepted as a valid answer, though not what we were expecting, was that additional packets could be caused by packet retransmission from loss.)

ii) Can the underlying network stack transmit *fewer* than ten data packets? If so, why?

Yes. Small writes can be coalesced into single data packets using Nagle's algorithm.

2C. [3 points] Suppose application A is using a datagram (UDP) socket to transfer data to application B on a remote host. Suppose application A calls *sendto()* ten times.

i) Can the underlying network stack transmit *more* than ten data packets? If so, why?

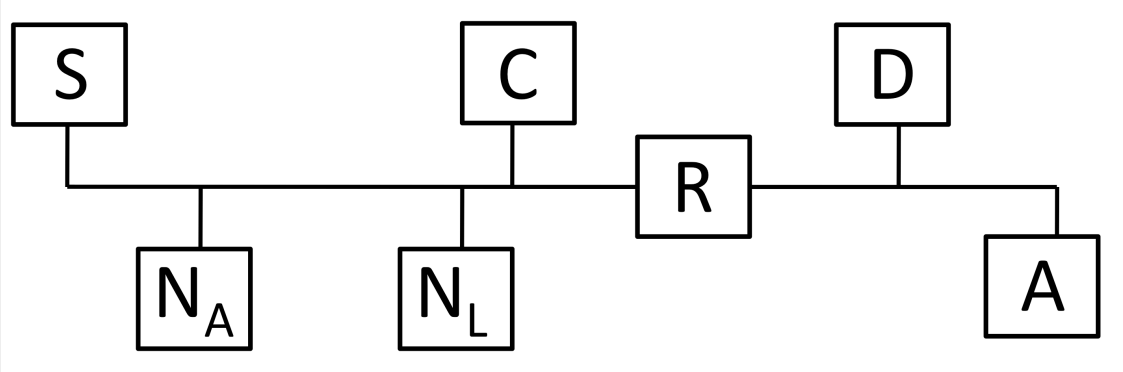
Yes. UDP packets larger than the network MTU will experience IP fragmentation.

ii) Can the underlying network stack transmit *fewer* than ten data packets? If so, why?

No.

QUESTION 3: Your Web Browser in action (17 POINTS)

In the topology shown below, machine A is a desktop client, D is a local DHCP server, R is A's gateway router, S is a Web server, and C is a Web cache. N_L is C's local nameserver, while N_A is the nameserver authoritative for S. Client A is configured to use Web cache C for all requests (assume that the Web cache resolves the name for any Web server and that the client is configured with the IP address of the cache). All wires/links are Ethernet segments.



Assume the following:

- All the machines were just booted and their associated caches (ARP, DNS, Web, persistent connections) are all empty
- http://S/index.html fits in a single packet
- Persistent HTTP connections are used among A, C, and S (i.e. you should assume that once any connection between these hosts is established it is never closed)
- Web caches respond to TCP requests that look like packet two in table 1 below (e.g., GET http://foo/bar/). They reply with the normal web cache contents.

3A. [12 points] The user on machine A, requests the web page http://S/index.html. The table below shows a number of messages sent/received in servicing this request (this is not necessarily a complete list of all packets). In addition, there are a few bogus packets that are never sent/received. The packets are not listed in temporal order.

ID	SRC	DST	SRC Port	DST Port	Protocol	Contents
1	C	DNS root		DNS	UDP	Query for S
2	A	C		Web cache	TCP	GET http://S/index.html
3	N_L	DNS root		DNS	UDP	Query for S
4	C	S		HTTP	TCP	SYN
5	C	S		HTTP	TCP	GET index.html
6	S	A	HTTP		TCP	index.html
7	A	Broadcast			ARP	Who is R
8	C	A	Web cache		TCP	index.html
9	N_L	C	DNS		UDP	Address for S
10	S	C	HTTP		TCP	index.html
11	A	Broadcast		DHCP	UDP	DHCP Request from A
12	N_L	N_A		DNS	UDP	Query for S

List the proper order of packets (in terms of their IDs) that are sent on the network:

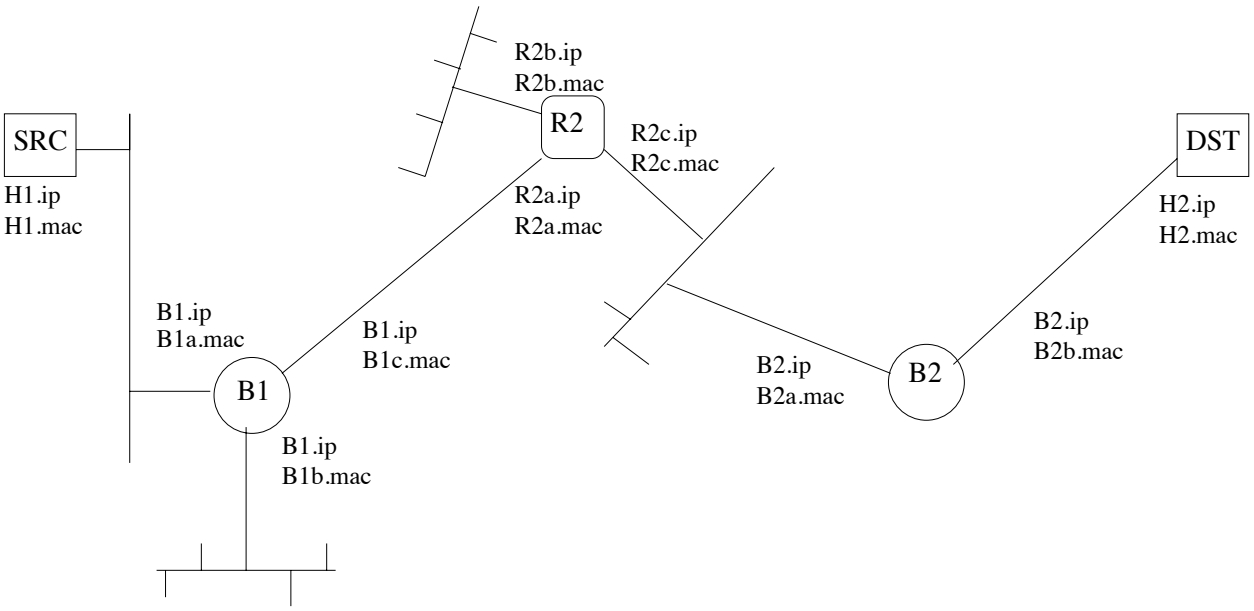
11, 7, 2, 3, 12, 9, 4, 5, 10, 8

3B. [5 points] Assume that the client A has no local Web or DNS cache and that cache C has no DNS cache. However, all other cacheable things are cached. On a subsequent request for http://S/ index.html, which of your items from your list in 3A would be eliminated? (Use the ID column to name the messages.)

11, 7, 3, 12, 9, 4, 5, 10

(The trickiest number was 9, because we said that C doesn't have a DNS cache. However, if it has the object for http://S/index.html cached, it can serve it directly there without going to server S, hence it never needs to do a DNS lookup on S in the first place.)

QUESTION 4: A routed and bridged network (14 POINTS)



Above is a picture of a network with 2 bridges and 1 router. Each interface is labeled with both an IP address and a MAC address. Imagine that host H1 is sending a packet to host H2. Please answer the following questions about this figure:

4A. How many (datalink) networks are shown above?

3 -- The networks hanging off of R2

4B. Just before the packet reaches bridge B1, what is its layer 2 destination?

R2a.mac

4C. Just before the packet reaches bridge B2, what is its layer 2 source?

R2c.mac

4D. Just after the packet leaves router R2, what is its layer 3 source?

H1.ip

4E. When H1 sends out an ARP query, what is the reply to that query?

R2a.mac has R2a.ip (or just "R2a.mac")

4F. Does the entry B2a.mac appear in B1's forwarding table?

No. B2a is on a different datalink network than B1.

4G. Will SRC's Ethernet device receive packets with destination R2a.mac? Why / why not?

This turned out to be a more confusing question than we intended, depending on how you interpreted the question. We accepted both "yes" and "no" answers here, but your explanation why had to be correct:

No -- because B1 is a (learning) bridge and any packets to R2a sent by nodes in on the "B1b.mac" or "B1c.mac" broadcast domains will be filtered by B1, so will only be re-broadcast out the B1c link.

Yes -- if the sender is in the same local broadcast domain (the "B1a.mac" domain) as SRC, then SRC will receive those packets.

Yes -- if B1 or R2 has just come online and B1 hasn't yet learned local MAC addresses of nodes, then it could forward packets out all interfaces.

QUESTION 5: Multiple choice *(16 points)*

Correct answers are marked in bold and underlined.

5A. Which best describes the Ethernet protocol? (Circle ONE)

A. Talk only if you hear no one else talking, but stop as soon as you hear anybody else.
B. Pass a ticket around and only talk if you are holding the ticket.
C. Raise your hand and wait till a moderator gives you permission to talk.
D. Every person is scheduled a time to talk.

5B. Which of the following is/are true about Ethernet: (Circle ALL that are correct)

A. CRC error detection, as used in Ethernet, cannot always detect if there is a frame error.
B. In the Ethernet CSMA/CD protocol, suppose that an adapter constructs a frame to send and then senses that the channel is busy. The adapter then enters exponential back-off.
C. Entries in an Ethernet bridge table must be configured by a network administrator.
D. An Ethernet adapter always passes all non-corrupt frames that it receives up to the network layer.

5C. Which of the following is/are true about Address Resolution Protocol (ARP) and learning bridges? (Circle ALL that are correct)

A. A learning bridge maintains state that maps IP addresses to hardware (MAC) addresses.
B. A learning bridge maintains state that maps MAC addresses to IP addresses.
C. A host's ARP table maintains state that maps IP addresses to hardware (MAC) addresses.
D. A host's ARP table maintains state that maps hardware addresses to IP addresses.

5D. Which of the following is/are true about a communications channel that uses time- division multiplexing? (Circle ALL that are correct)

A. There may be times when the channel is idle, even if a sender has data to send on the channel.
B. The channel requires the sender's and receiver's clocks to be closely synchronized.
C. Data in the channel could experience variable delays due to queuing.
D. In times of high utilization, a sender could be completely denied access to the channel.

5E. Which of the following is/are true about increase/decrease policies for fairness and efficiency in congestion control? (Circle ALL that are correct)

A. Additive increase reduces fairness.
B. Additive increase improves efficiency.
C. Multiplicative increase improves fairness.
D. Multiplicative decrease improves fairness.

5F. Which of the following is/are true about DNS? (Circle ALL that are correct)
A. A query for an A record may return multiple IP addresses in the response.
B. A query for an NS record may return multiple IP addresses in the response.
C. A short TTL on an NS record reply runs the risk of increasing traffic at the root or GTLD nameservers.
D. A short TTL on an A record reply runs the risk of increasing traffic at the root or GTLD nameservers.

5G. Which of the following is/are true about web caches? (Circle ALL that are correct)
A. A web cache, or proxy server, is a network entity that satisfies HTTP requests from clients.
B. A web cache is both a client and a server at the same time.
C. HTTP does not explicitly support caching or cache consistency.
D. All HTTP objects are cacheable.

5H. Which of the following is/are true about persistent HTTP connections? (Circle ALL that are correct.)
A. Persistent HTTP allows a server to track the client's requests through a persistent session.
B. Only one TCP connection must be opened for downloading a "page", if that page does not include any embedded objects served by other servers.
C. Persistent HTTP shows the greatest performance advantage over nonpersistent HTTP when downloading a page with *large* objects.
D. When the server has finished sending data for all objects referenced by the initially requested page, the server closes the connection.

For this question, we also marked correct if one circled A, as the question was perhaps overly confusing (and virtually everybody got the answer wrong). Strictly speaking, persistent HTTP does not establish application-layer sessions, that's typically done through cookies. This is the case because "application-level" sessions exist independently of a single TCP connections. It can more obviously be seen through the use of web proxies -- a web proxy can/will multiplex requests from different clients over a single TCP connection with the server. That said, while 5G and 5H should be answerable given assignment 2, this particular question (5H-A) might have been difficult without lecture 16, so we decided to effectively give everybody credit.

QUESTION 6: Short Answer *(22 points)*

6A. [2 points] Explain the principle of soft state. Provide two examples of a network protocol or technology that abides by soft state.

Soft state refers to some type of stored state that is not "hard" (i.e., persistent, consistent, explicit, etc). In the context of the applications we saw, soft state is often only stored temporarily by some entity, often for only some explicit or implicit time period, and it must often be constantly refreshed to survive network changes or failures or persist over time. Existence of soft state should not be critical from receiving correct service; that is, it is often an optimization.

Accepted answers: ARP tables, DNS caches, DHCP addresses that persist over leases, ...

6B. [2 points] List two protocols that use exponential backoff, and describe what/where problems would occur if each protocol did not backoff.

TCP congestion control uses exponential backoff (multiplicative decrease). If it did not backoff, congestion would build up at routers, and queues would continue to be full and continue to drop packets, leading to low throughput.

Ethernet CSMA/CD uses exponential backoff. If it did not backoff, collisions would continue to occur on the shared medium, leading again to low goodput.

6C. [3 points] List three possible explanations for why host A would need to retransmit the SYN packet as part of establishing the TCP connection with B.

SYN packet from A to B dropped by router on path.
SYN packet dropped by B because B's receive backlog is full.
SYN-ACK packet from B to A dropped by router on path.
SYN-ACK not received by A before timeout occurs.
SYN dropped by firewall or NAT in front of B.

6D. [1 point] Give one reason that DNS lookups are run over UDP rather than TCP

DNS is a simple query/response protocol, thus avoids connection setup (at least 1 RTT) for TCP and its state overhead. This is especially true as DNS queries and responses are typically small and can thus often fit in a single packet. Because its a connectionless protocol, also more amenable to anycast IP addresses for DNS servers, useful for failover. UDP headers are also smaller than TCP headers, which reduced bandwidth overhead for queries.

6E. [2 points] Give two reasons that sites use Network Address Translators (NATs):

Allows multiple private addresses behind single public (dedicated) address, which is very helpful if sites lack a sufficient number of public IP addresses (e.g., "space" pressure). In general, NATs have helped "scale" IPv4.

NATs provide privacy about internal deployments.

NATs can allow easily management of internal devices, so that machines can be internally renumbered without changing their external (public) addresses.

If explicit forwarding rules aren't configured in NATs, they often provide some security mechanisms by playing the role of firewalls, e.g., only allowing TCP connections to be initiated from internal nodes, as opposed to allowing external machines to connect to internal machines.

...

6F. [3 points] When a packet is going from a local-area network onto the wider Internet, what header fields might a Network Address Translator (NAT) change that is deployed on path?

source IP address, source port, IP checksum, transport (TCP/UDP) checksum

6G. **[**3 points] How does explicit congestion notification (ECN) differ from traditional ways of detecting congestion on Internet paths? Describe one benefit for using ECN and one reason it might not widely be used today.

ECN signals to senders that congestion has occurred by marking packets explicitly, as opposed to signalling by "dropped" packets. This allows end-hosts to react to congestion without unnecessary loss, especially useful, for example, when routers might be using RED, or for interactive application-level protocols such as telnet and web browsing that are very sensitive to the delay caused by loss and retransmission. ECN is not widely used today, however, in that it requires that both the sender, recipient, and network elements (routers) all support ECN, so it creates a deployment challenge.

6H. [4 points] The OSI model is traditionally seen as a 7-layer stack consisting of the following. For several of the layers, give one or two examples of protocols that are associated with each layer:

Application	**2 examples: FTP, HTTP, SMTP, telnet, SIP, ...**
Presentation	
Session	
Transport	**2 examples: TCP, UDP, DCCP, ...**
Network	**1 example: IPv4, IPv6**
Data link	**2 examples: Ethernet, PPP, token ring, FDDI, 802.11**
Physical	**1 examples: We accepted both physical techologies (twisted pairs, CAT5, fiber, ...) and encodings (Manchester, NRZ, ...)**

6I. [1 point] What's the first name of your favorite TA? Why?

Jeff (22 votes) and Wyatt (15 votes)

6J. [1 point] Describe one thing you would like to see changed in the class. (Only wrong answer is "nothing".)