Midterm Exam
CMPSCI 453: Computer Networks
Fall 2010
Prof. Jim Kurose

Instructions:

- **Please use two exam blue books** – answer questions 1, 2 in one book, and the remaining two questions in the second blue book.
- Put your name and student number on the exam books NOW!
- The exam is closed book.
- **You have 80 minutes** to complete the exam. **Be a smart exam taker** - if you get stuck on one problem go on to another problem. Also, don't waste your time giving irrelevant (or not requested) details.
- The total number of points for each question is given in parenthesis. There are 100 points total. An approximate amount of time that would be reasonable to spend on each question is also given; if you follow the suggested time guidelines, you should finish with 10 minutes to spare. The exam is 80 minutes long.
- Show all your work. Partial credit is possible for an answer, but only if you show the intermediate steps in obtaining the answer.
- Good luck.

Question 1: ``Quickies'' (32 points, 20 minutes)
Note: Four points each (pretty much right or wrong)

Answer each of the following questions *briefly*, i.e., in at most a few sentences.

a) Suppose there are 5 users whose traffic is being multiplexed over a single link with a capacity of 1 Mbps.
   1) Suppose each user generates 100 kbps when busy, but is only busy (i.e., has data to send) 10% of the time. Would circuit-switching or packet-switching be preferable in this scenario? Why? *Answer: Here, circuit switching is preferable since each of the users will each get a dedicated allocation of 100 kbps.*
   2) Now suppose that the link capacity is still 1 Mbps, but the amount of traffic each user has to send when busy is increased to 1 Mbps, and that each of the 5 users still only has data to send 10% of the time. Would circuit-switching or packet-switching be preferable in this scenario? Why? *Answer: Here, packet switching is preferable. We can not allocate 1Mbps per user in circuit-switching mode. Packet switching will work well since the aggregate average traffic rate is 0.5 Mbps and the link is a 1 Mbps link*

b) What is the purpose of the IF-MODIFIED-SINCE field in an HTTP GET message? *Answer: The HTTP server will only send the requested content if the content has been changed <u>after</u> the date specified in the IF-MODIFIED-CINE field.*

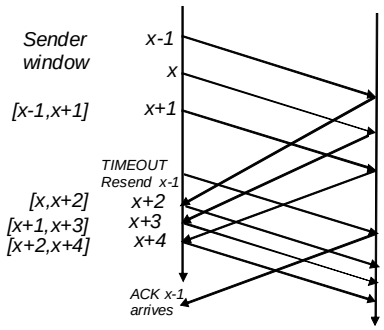
c) Suppose a TCP SYN message is sent from a client with IP address 128.119.40.186 and client port number 5345 to a server with IP address 41.123.7.94 and server port number 80 (HTTP)
   1) Once the TCP connection has been established, what will be the client-side IP address, client-side port number, server-side IP address and server-side port number of the TCP segment carrying the HTTP GET message. *Answer: exactly as specified in the problem statement.*
   2) Will the TCP SYN message and the HTTP GET message be directed to the same socket at the server? Explain in one or two sentences. *Answer: NO. The GET will be directed to the new socket that was created when the TCP SYN messages was accepted (i.e., the socket returned from the wait on the .accept() on the welcoming socket). Note that the TCP SYN and the GET will both be addressed to port 80 on the server, however.*

d) In *network-assisted* congestion control, how is congestion in the network signaled to the sender? *Answer: the routers may set bits in passing packets, and may send messages to the source or destination to indicated congestion.* Name one protocol that uses a network-assisted approach. *Answer: ATM ABR uses a network-assisted approach.* In *end-end* congestion control, how is congestion in the network signaled to the sender? *Answer: congestion is inferred at the end hosts (sender or receiver, either as a the result of packet loss or increased delay).* Name one protocol that uses an end-end approach. *Answer: TCP uses an end-end approach.*

e) Briefly describe (in a few sentences) how *<u>caching</u>* is used in the DNS (note: you are *not* being asked to describe the entire DNS). *Answer: a host will maintain a cache of recent DNS-translated address/name pairs. If a name is found in the cache, the DNS system will not be consulted to perform the mapping.* Are values returned from a DNS

cache always guaranteed to be up to date? Explain. *Answer: NO. Cached values have a time-to-live value and will remain in the cache until they time out. If the name/address pair is changed in the DNS, the cached value will not change, and the current mapping will only become known after the old mapping times out of the cache, and a new mapping is retrieved from the DNS.*

f) Consider a sliding window reliable data transfer protocol (e.g., Go-Back-N, or Selective-Repeat). Suppose the sender window contains sequence numbers in the range *[x,x+N]*. Is it possible for the sender to receive an ACK that is for a sequence number that is less than *x? Answer: yes see diagram.* If so, sketch a scenario showing how this can happen. If this can not happen, explain why this can not happen.

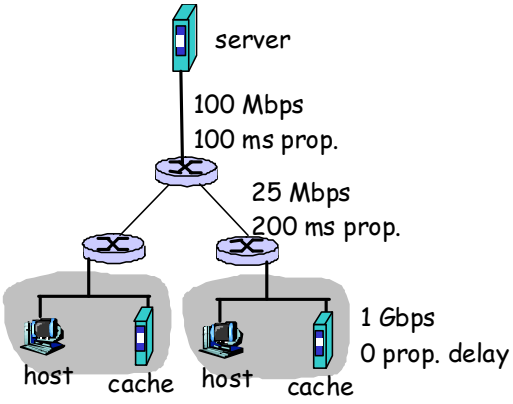


g) Does the Internet checksum always detect errors in the transmitted segment? Explain your answer in a sentence or two. *Answer: No. For example if two 16-bit word values are swapped, this would not be detected since the sum is unchanged.*

h) What is meant by *demultiplexing* a protocol data unit up to an upper-level protocol? *Answer: this refers to passing the decapsulated data unit up to the appropriate higher level protocol. This is done by looking at the upper-layer protocol field.*

Question 2: Delays, Throughput and Caches (18 points, 10 minutes)

Consider the scenario in the figure below in which a server is connected to a router by a 100 Mbps link, with a 100 ms propagation delay. That router in turn is connected to two routers, each over a 25 Mbps link with a 200 ms propagation delay. A Gbps link connects a host and a cache (when present) to each of these routers; this link, being a local area network, has a propagation delay that is essentially zero. All packets in the network are 10,000 bits long.



a) 4 points. What is the end-to-end delay from when a packet is transmitted by the server to when it is received at a host? Assume that there are no caches, that there is no queueing delay at a link, and that the node (router) packet-processing delays are also zero. *Answer: If all packets at 10,000 bits long, it takes 100 usec to send the packet over a 100Mbps link, 400 usec to send over a 25 Mbps link, and 10 usec to send over a gigabit link. The sum of the three link-transmission times is thus 510 usec. The sum of the propagation delays is 200+100=300 msec.* ***Thus the total end-end delay is 300.510 msec.***

b) 3 points First assume that client hosts send requests for files directly to the server (i.e., the caches are off). What is the maximum rate at which the server can deliver data to a single client, assuming no other clients are making requests. *Answer: 25 Mbps, the bottleneck link speed.*

c) 3 points Again assume that only one client is active, but now suppose the caches are HTTP caches and are turned on. A client HTTP GET is always first directed to its local cache. 50% of the requests can be satisfied by the local cache. What is the maximum rate at which this client can receive data in this scenario? *Answer: We assume that requests are serially satisfied. 50 % of the requests can be delivered at 25 Mbps and 50% of the requests can be delivered at 1 Gbps. So the average rate is 512.5 Mbps.*

d) 4 points Now suppose that the clients in both LANs are active and the HTTP caches are on, as in c) above. 50% of the requests can be satisfied by the local cache. What is the maximum rate at which each client can receive data, in this scenario? *Answer: the 25 Mbps remains the bottleneck link, which is not shared between clients. So the answer is the same as c) above. Note that we assume that the 100Mbps is shared at a fine grain, so that each client can get up to 50Mbps over that link.*

e) 4 points . Now suppose the 100 Mbps link is replaced by a 25 Mbps link. Repeat question d) above in this new scenario. *Answer: The two clients must now share the 25 Mbps bottleneck link, each getting 12.5 Mbps. 50 % of the requests from a client are delivered at 12.5 Mbps and 50% of the requests can be delivers a 1 Gbps. So the average rate is 506.25 Mbps.*

Question 3: TCP potpourri (20 points, 15 minutes)

Answer questions a) – d) below briefly, in just a few sentences. Question e) will take some more time and thought and require a longer answer.

a) (3 points) What is the purpose of the receiver-advertised window in TCP? *Answer: this allows the receiver to tell the sender how much unacknowledged data can be in flight.*

b) (3 points) Identify 6 fields in the TCP header, and give a one-sentence description of each of their uses. *Answer: souce port, dest port, seq num, ack num, checksum, receiver window, options, SYN, FIN, ACK, urgent pointer, PSH.*

c) (3 points) Consider two TCP sessions that must share a link's bandwidth. One of the TCP connections has been running for quite some time and has built up a large TCP sending window. The second connection then starts up with an initially small window. Long term, what will be the relative throughput achieved by these two TCP sessions? Explain. *Ans: We saw that TCP will cause two senders to eventually fairly share the links bandwidth, so each will eventually have the same size window (assuming their RTT is the same).*

d) (4 points) Consider a TCP session and a UDP session that must share a link's bandwidth. Of course, both sessions would ideally like to send as fast as they can. Long term, what will be the relative throughput achieved by these two sessions. Explain. *Ans: Since UDP can send as fast as it wants, it can use all of the bandwidth (e.g., in the limit that it sends infinitely fast, as soon as buffer space becomes free in a router, that buffer will be filled by a UDP segment. TCP segments will always be lost, causing TCP to keeps its window at 1 segment, which when sent is always lost.*

e) (7 points) Suppose we want to modify TCP so that it counts the number of segments that are lost in transit. What are the difficulties of doing this in TCP? *Answer: the key complications here are (i) that ACKs are cumulative (and so the sender may never see some packets individually acked); (ii) premature timeouts result in segments being resent that were never lost in the first place, and (with acks being cumulative) one cant tell how many of the retransmitted segments are lost.*

- *Because ACKs are cumulative, if the sender gets no ACK for x but gets an ACK for x+1, it doesn't know if x was received and the ack for x was lost, or if x was lost but a later retransmission of x was received.*
- *A lack of an ACK for x could just mean that the ACK for x is delayed. Suppose x is retransmitted. The sender may <u>never</u> see an ACK for the retransmitted x because of cumulative ACKs for segments with a higher sequence number than x.*
- *Because of the ACK-every-other segment behavior of TCP receivers, an ACK may never be generated for x is x+1 is received soon after x.*

Problem 4. A Reliable Balance Updating Protocol (30 points, 25 minutes)

Consider the following scenario. A sender is called locally and passed updates to increase or decrease a stored value that is maintained at a remote receiver (you can think of these increases and decreases as deposits and withdrawls from a bank account whose balance is maintained at the receiver). The sender must communicate with the receiver so that the balance value stored at the receiver is consistent with the value at the sender side, given the increases or decreases being requested at the sender side. (Of course, the sender and receiver values can never be perfectly synchronized since there is a delay between the sender and receiver).

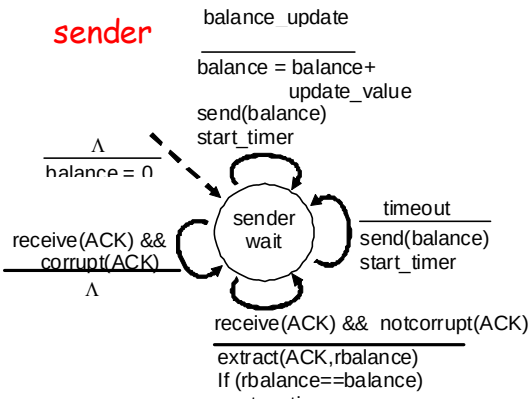
The sender and receiver communicate with each other over channels that can corrupt and lose packets, but will not reorder packets. The delays of the channel are unknown and can vary over time.

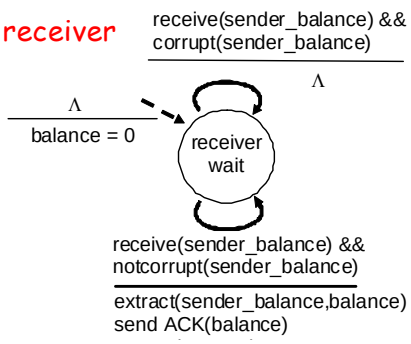
The sender is invoked by a call to `balance_update(update_value)` which is analogous to the `rdt-send(data)` sender-side invocation we saw in the *rtd* protocols; `update_value` is a positive or negative number reflecting the increase or decrease (analogous to deposit/withdrawl) to the balance.

The receiver should track the value of a variable `balance` (which starts at 0), which tracks the value of the balance, and can be positive or negative. The updates at the receiver do not need to be processed in the order in which there were initially received at the sender, but eventually the value of `balance` at the receiver must equal the sum of the increases/decreases made at the sender via `balance_update(new_value)`, and of course each update request made at the sender must be reflected only once at the receiver.

a) Design a protocol between the sender and receiver, operating over the lossy and error-prone channel, to ensure that the receiver's value is as close as possible to that resulting from the sequence of increases and decreases made at the sender. Specify your protocol using finite state machines. (*Hint:* there are many possible correct solutions here; some are easier than others. You might want to think a bit about how this balance-updating problem differs from the reliable in-order data delivery problem that we studied in class.). *Answer: here is my solution. Note that there are separate FSM for sender and receiver (!).*

b) For each state in your FSM, give a *one sentence* description of what it means when the sender or receiver is waiting in that state. *Answer: there is just one wait state in each.*





The sender waits for a call from above (with an update), a timeout (in which case it will resend), or arrival of a message (which is corrupt or not). The receiver just waits for a message from the sender (which is corrupt or not). Note that I didn't use sequence numbers here since the basic idea is that the sender is simply going to keep updating values, and will always send the most recent update. By putting updates at the sender, the receiver simply needs to deliver up correctly received balance values.

c) For each type of message sent between sender and receiver, specify the fields in each type of message and the values/meaning of each field. *Answer: the sender sends the current balance (say a floating point) and will include a CRC to detect bit errors. The receiver sends an ACK with the current value of the balance.*

d) Suppose now that the medium can reorder messages. Give a short description (one paragraph) of how you would modify you protocol (if at all) to accommodate this. *Answer: if the medium reorders the balance at the receiver will always correspond to the balance at the sender at some point in time. If we need to the balance to be delivered sequentially at the receiver, we'd need to add sequence numbers that are large enough so that wrap around wouldn't be a problem.*

e) Suppose now that the link connecting the sender and receiver has a long propagation delay. Describe in a few sentences how you would modify (if at all) your protocol in a) to accommodate long-delay links. You do not have to provide an FSM, just briefly describe (in words) any changes you would make. *Answer: for my solution, I wouldn't change anything since the sender is always sending the most recent value. If one used a stop-and-wait-sequence-numbered-approach like we used in rdt 3.0, then one would want to add pipelining so that the sender could have multiple updated balanced in transit.*