# 6.896 Topics in Computer Networks: Midterm Quiz

Name: Solutions

Email: hari@lcs.mit.edu

November 4, 1998

This test has **17 questions** and you have **90 minutes** to answer them. Keep your answers *brief, legible, and to the point,* answering them in the space provided. Draw pictures if you think they will clarify your answers. You may use the empty space at the back of each page for scratch work. **You will get partial credit for partially correct answers. Attempt as many questions as you can.**

**Maximum score: 60 points.**

**Good luck and relax!**

1. **Philosophy**

   (a) What does "fate sharing" mean? (2 points)

   *The idea that all the state is pushed as far as possible to the end-points and away from the inside of the network, so the only hosts that share fate are the communicating peers. Then, each peer shares the fate of the other; if one crashes, then no communication can occur.*

   (b) Give two examples of how *unnecessary* fate sharing occurs in the Internet (or from the papers you have read in this class so far). (2 points)

   *There are several. DNS. Mobile IP's home agent. Split-connection TCP (I-TCP), etc.*

2. **TCP and IP**

   (a) TCP has an end-to-end checksum that covers part of the IP header, in addition to the TCP header and data. When the receiver receives a data segment whose checksum doesn't match, it can do one of two things: (i) discard the segment and send an ACK to the data sender with the cumulative ACK field set to the next in-sequence byte it expects to receive, or (ii) discard the segment and do nothing else.

   Is one action preferable to the other (or are they both equivalent)? Why? (2 points)

   *(ii) is the appropriate action. If the checksum is wrong, the receiver doesn't know whom to send the ACK to with certainty, since the IP addr and TCP ports might be the corrupted fields.*

   (b) The IP layer performs the functions of fragmentation and reassembly of datagrams. Yet, routers don't perform reassembly. Why not? (2 points)

   *This is an example of the end-to-end principle in action. In addition, parallel paths would be precluded if routers did perform reassembly.*

3. **Congestion Management**

   (a) DECbit uses a 7/8 window reduction factor while TCP reduces its window by 1/2. Does this mean that DECbit's factor too aggressive? Or is TCP's too conservative? Explain your answer. (2 points)

   *DECbit is an avoidance scheme, since it flags congestion bits even before queue overflow occurs. For this, a larger multiplicative factor is reasonable. TCP reacts only after a loss, i.e., only after a queue overflow. In slow start, if the sender loses packets in a window, it only knows that half the window worked, motivating the more drastic one-half factor.*

(b) Using the $(x_1, x_2)$ phase plots *a la* Chiu and Jain, show that an additive increase / multiplicative decrease congestion management scheme does not monotonically converge to fairness with non-synchronized feedback (even if round-trip times are equal). (2 points)

*In the $(x_1, x_2)$ plane, suppose you're at overload and $x_1 > x_2$. Suppose $x_2$ alone reduces because of non-synchronized feedback. This reduces fairness.*

(c) Are RED gateways (say, in packet marking mode) more or less likely (compared to FIFO gateways) to lead to synchronized feedback to all sources, upon experiencing congestion? (2 points)

*In general, less likely because RED (to first order) marks/drops packets roughly proportional to their bandwidth use. Drop-tail gateways don't have any such biases, treating every flow with equal disdain!*

(d) Instead of the standard drop-tail scheme, some people have proposed *drop-from-front* as an algorithm to run at a congested gateway to decide which packet to drop. Here, packets at the head of the queue are dropped instead of incoming tail packets. Why might this be better than drop-tail? (1 point)

*Feedback reaches the sender sooner about loss and the state of congestion.*

4. **One-file HTTP**

(a) Consider an HTTP download of a small file (an *index.html*) of size $S$ from a Web server. The file is small enough so that the connection remains in slow start, and assume that no losses occur during the download. How many round-trips does this take, using TCP as the transport. Assume that the request for the file fits in one packet. Assume that the TCP does not implement delayed ACKs (i.e., it ACKs every segment) and that the maximum segment size is $m$. (2 points)

*One round-trip for connection establishment; then one round-trip to send request and get first segment ($m$ bytes). At this point your window size is two. The number of round-trips before you get all the $S$ bytes works out to $1 + \log_2[(S/m)]$, where $[x]$ is the smallest integer $\geq x$.*

(b) How does your answer change with a delayed ACK receiver that sends an ACK for every other successfully received segment? (An approximate answer will suffice, you don't need to write the precise expression) (1 point)

*The $log_2$ becomes $log_{1.5}$.*

5. **Fair queueing**

Non-preemption in the implementation of fair queueing violates the precise definition of bit-by-bit round-robin service. Demonstrate this using a simple example. (2 points)

*Consider a very large packet in service and a smaller packet now arrives, and it has an earlier finish round.*

6. **Is this really fair queueing?**

   In the fair queueing paper we read in class, we saw recurrence relationships for $F_i^\alpha$, the finishing round number. To simulate bit-by-bit round robin, the router transmits packets based on the earliest $F_i^\alpha$.

   Instead, consider a different scheme that sends packets based on earliest *starting* round.

   (a) Write the corresponding recurrence relationships for $S_i^\alpha$. (1 point)

   $S_i^\alpha = \max((S_{i-1}^\alpha + P_i^\alpha), A_i^\alpha)$.

   (b) Does this scheme actually come close to emulating bit-by-bit fair queueing? Explain your answer. (2 points)

   Yes, it does. If all the flow queues are full, then it's easy to see that this is in fact very similar to regular fair queueing. When a packet hits an empty queue, it still holds because $F_i^\alpha = S_i^\alpha + P_i^\alpha$. I.e., as long as packet sizes are reasonable, this does in fact emulate bit-by-bit round-robin quite well.

7. **TCP retransmissions**

   (a) Could the sender retransmit as soon as it received the first duplicate ACK in this scheme? Why (not)? (1 point)

   *Nope, needs to guard against packet reordering in the Internet.*

   (b) Suppose that in addition to its cumulative ACKs, TCP also implemented selective ACKs, where the receiver informed the sender which out-of-order segments caused each duplicate ACK. Give two reasons why TCP the sender *still* needs to implement timeout-driven retransmissions (2 points)

   *(a) In a window of size $W$, at least $W - 2$ packets are lost. Not enough duplicate ACKs arrive. This is all-too-common for Web workloads where $W$ is often quite small.*

   *(b) Many ACKs/SACKs are lost.*

8. **Fat pipe networks**

Many people have claimed that TCP performance over very high-bandwidth networks (e.g., multi-gigabit fiber or optical) and satellite networks will be very similar in the Internet. The claim is that while the first network has very high bandwidth and low delay, the second has high bandwidth and very long delay, and the bandwidth-delay products in the two cases are roughly the same. Let's assume that the bandwidth-delay products are in fact the same (not an unrealistic assumption in many cases).

Do you think the TCP performance over the Internet for two different connections over these two technologes will in fact be similar, assuming similar rates of loss due to congestion? (Assume that all losses are due to congestion). Explain your answer. (2 points)

*No, throughput depends on the RTT for TCP because of the way its congestion avoidance works. The longer rtt connection will lose in performance to the shorter one, even if both pipe sizes are roughly the same. This is because TCP in congestion avoidance increases its window by one every round-trip, and the longer rtt connection will take longer to build up its window between losses.*

9. **Exterior Routing Protocols**

(a) What is the main difference between EGP (the old Exterior Gateway Protocol) and BGP (the Border Gateway Protocol)? What problem does this difference address (i.e., why bother doing something more complex in BGP)? (2 points)

*Path vectors. Now, the backbone isn't constrained to be a tree as it used to in NSFNET, since loop avoidance is now easy to do.*

(b) Name one advantage and one disadvantage of running BGP over TCP, rather than over UDP. (2 points)

*Reliability is taken care of. But performance is sluggish and awful when the network is congested because TCP invokes congestion control.*

10. **Landmark routing**

Unlike Tsuchiya's design, could you use a link-state abstraction to build a landmark routing scheme? Explain your answer. (2 points)

*No, the idea with landmarks is not to have global topological knowledge like the link-state protocols have.*

11. **Optimal wavefronts**

In wavefront switch scheduling algorithms, input to output port matching is done in "rounds" based on wavefronts progressing from the $(1, 1)$ (input,output) pair to the $(N, N)$ pair. In each round, all the matches are done in parallel, since no output conflicts can occur.

(a) Construct an example where this algorithm leads to a sub-optimal scheduling of inputs to outputs, assuming you want to maximize switch throughput. (2 points)

*Consider a 2-by-2 switch with the following allocations.*

| *Port #* | 1 | 2 |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 0 |

*In this situation, wavefront pairs input port 1 with output port 1 and matches no other ports. The optimal pairing is, of course, 1 to 2 and 2 to 1.*

(b) This algorithm can be made more efficient if you segment the switch grid into smaller chunks and act on them in parallel. What is the optimal segmentation? Prove that your answer is correct. (For simplicity, you can assume that $N$ is a perfect square or a power of 2, or anything that makes your proof convenient). (2 points)

*If you break the $NXN$ grid into contiguous squares of $mXm$, then some allocations can be performed in parallel. That is, you can perform a "higher-level" wavefront algorithm starting from the top left square and proceeding downwards like the conventional wavefront. Now, within each $mXm$ square, it takes time $2m - 1$ to come up with a wavefront schedule. This is embedded within a higher-level wavefront that propagates at the square level, which takes time $2N/m - 1$ since there are $(N/m)^2$ squares in all. Thus, the total time is equal to $2m - 1 + 2N/m - 1$. This is minimized when $m = \sqrt{N}$.*

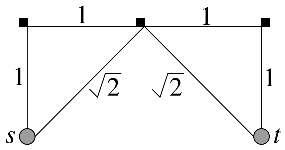
12. **Minimum-energy routing**

Consider the network topology shown in the figure above (the edge weights are distances; assume that power drops off as $1/r^2$).

(a) How many hops does the minimum energy route between $s$ and $t$ take? (1 point)

*4 hops.*

*Looking through the answers, I realize this question caused lots of confusion. Here's my argument. The total energy consumed going from $s$ to $t$ along the four hop "right-angled" path is equal to $P * 4\delta$ where $\delta$ is the propagation time to traverse unit distance and $P$ the power needed to achieve successful transmission over unit distance. The energy needed to*




*go via the two-hop path is $2P * 2 * sqrt(2)\delta$, which is a larger. I realize that the descripion in Shepard's paper threw most people off from the "first-principles" definition. On the other hand, if you were to implement this scheme, you would probably not know the distance too well and just assume that propagation time is the same to all neighbors, in which case both paths are equivalent.*

(b) If it takes power $P$ to send a message 1 unit distance in this network, how much more interference does the shortest path cause over the minimum-energy route to points far away? (1 point)

*Shortest path is a factor of two higher in terms of maximum interference caused.*

13. **Multicast**

(a) What primary problem with DVMRP do PIM-SM and CBT address? (1 point)

*Poor scaling caused by periodic global flooding, $O(S * G)$ router state and state in routers with no members downstream. These are especially relevant for sparse groups.*

(b) Are reverse path checks (performed in DVMRP routers) useful in CBT? (1 point)

*No. Packets are sent along core tree.*

14. **Wireless multicast**

    You have just deployed a wireless LAN in the LCS building, with all the nodes (wireless interfaces of the multiple base stations and the mobile hosts) on the same subnet (for simplicity). Your mobile hosts now try to join the MBone and get IP multicast traffic using IGMP, but it doesn't always work.

    (a) Explain clearly why you see this whimsical behavior. (2 points)

    *The problem is caused by the way IGMP querying works. If there are multiple routers to a subnet, then one of them arbitrarily gets elected (based, e.g., on lowest IP address) to the the querier for the subnet. Now, in the wireless case, this means that one of the base station wins this election. But this base station won't in general cover all of the cells in the building, so depending on where you move in the building, you may or may not get multicast traffic from the outside world.*

    (b) Suggest a way to fix this. (2 points)

    *Just turn off the querying algorithm between the mrouters on the base stations. This is sub-optimal, but will work.*

    *Note: For this question, I gave points for all reasonable answers, provided they highlighted a relevant problem.*

15. **Fast classification**

    The Waldvogel *et al.* paper describes algorithms for fast prefix matching to speed packet forwarding. Suppose you wanted to apply their algorithms to the problem of *packet classification.* In this problem, you want to decide whether or not to forward a packet, driven by a set of rules about the contents of the incoming packet header (source and destination addresses and ports and the protocol field). (This is pretty much how firewalls work, so you can think of what you're trying to do as a step in making a fast firewall).

    You're doing fine with rules that involve prefix matches (e.g., all packets from network 18.31.0.0 are disallowed), but then you see a rule that involves a *range* (e.g., all packets with destination port numbers between 1000 and 5000 are disallowed). Can you figure out how to write a range as an efficient union of prefixes, so that you can leverage the prefix matching algorithms to do fast range checks? How many distinct prefix matches does a single range query become in the worst case for 16-bit port numbers? Assume that all information is stored in bits, since the prefix matching scheme works with binary strings; e.g., all port numbers in the range 0 through 1023 can be written as the prefix 000000*. (4 points)

*Some examples make it easy to answer this question. The range* $[8, 11]$ *is trivial to express as a prefix—it is just* $10*$. *The range* $\geq 2048$ *is a little more involved—it is the union of* $00001*, 0001*, 001*, 01*, 1*$.

*To convert an arbitrary range* $R$*, first pick an enclosing binary range* $[0, 2^k]$ *for some* $k$. *You can split* $R$ *into at most two contiguous sub-ranges that lie in* $[0, 2^{k-1} - 1]$ *and* $[2^{k-1}, 2^k]$. *Each of these can be split into a logarithmic number of prefix rages by constantly halving the range. Then, at each stage, the halving contributes one more prefix range. Thus, by breaking up the given range* $R$ *into a union of ranges in the form shown above, you can express any sub-range of* $[0, 2^k]$ *as a union of at most* $2k$ *prefixes. Thus, in the worst case you need about 32 (*$= 2 \log 2^{16}$*) prefix matches for 16-bit port numbers.*

16. **RLM**

(a) In McCanne *et al.'s* RLM protocol, explain why shared learning isn't used by receivers to advance to higher layers? (2 points)

*Shared learning is necessarily conservative because the receiver who performed the join experiment may have no links in common with another. So just because the latter didn't see any resulting losses doesn't mean it can happily add a layer.*

(b) In RLM, explain how a receiver might actually be experiencing congestion, but not react to it (i.e., not drop a layer in response). (2 points)

*Same situation as above, but this time a receiver thinks a sequence of losses was caused by a failed join-experiment when in fact it had no links in common with the experimenter.*

17. **Graphs.** Schematically sketch the following graphs, clearly labeling the axes. If you can derive closed-form analytical expressions for the curves you're sketching, write them out.

(a) Each hop in a multi-hop wireless network has a *small* packet error probability, $p$, after link-layer error correction has been done. Plot the overall path error probability $P(x)$ as a function of $x$, the number of wireless hops between sender and receiver. (2 points)

$P(x) = 1 - (1 - p)^x$ *which for small* $p$ *works out to* $P(x) = px$. *Straight line through origin, slope* $p$. *I.e., the path loss probability is roughly the sum of the individual link loss probabilities.*

(b) Consider a stock ticker application, where getting a message from the sender in $\leq d_1$ ms is best (corresponding to a maximum utility of $U_m$), and getting a message between $d_1$ ms and $d_2$ ms leads to a continuous *linear* performance degradation from $U_m$ down to zero. All messages delayed by more than $d_2$ ms are useless to the application (zero utility). Sketch the utility function as a function of the rate available to it, paying special attention to the shape of the curve in the different rate regimes. (3 points)

*When the $rate \leq S/d_2$, $U(r) = 0$. When $rate \geq S/d_1$, $U(r) = U_m$. (Here, $S$ is the packet size). The interesting stuff happens in between.*

*In the delay domain, the curve in $(d_1, d_2)$ is a straight line with negative slope. The equation of this line is easy to obtain, it is $U(d) = U_m/(d_2 - d_1)(d_2 - d)$. Now, to write the utility as a function of the rate, $r$, notice that $d$ is simply $S/r$. So, we have $U(r) = U_m/(d_2 - d_1)(d_2 - 1/r)$, which is a constant minus a term that looks like $K/r$ for some constant $K$. So, the curve looks like a shifted negative $1/r$ hyperbola in the rate range $(S/d_2, S/d_1)$. The curve is* concave *in this range.*

In a network dominated by this application, do you think admission control will help increase the aggregate utility? Explain your answer. (1 point)

*Yes admission control will help because there are rates (below $S/d_2$, where $S$ is the packet size), where the curve is not concave (in fact it is zero). In this regime, the application is inelastic.*