Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.829 Fall 2002 **Problem Set 2** September 26 2002

This problem set has six questions, most with several parts. Answer them as clearly and concisely as possible. You may discuss ideas with others in the class, but do not copy solutions from them or from anywhere else. Turn your solutions in by **Thursday, October 17 2002** in class.

The first two problems involve running `ns` simulations.

1 TCP over Long Fat Networks (LFNs)

The goal of this problem is to familiarize yourself with the basics of `ns` while examining the interaction between TCP, LFNs and routers. LFNs (pronounced "elephants") are networks with "fat pipes," where the bandwidth-delay products are large.

Two about-to-disappear .com's AlphaWidgets.com and BetaGadgets.com have decided to connect their networks to divert each others' dwindling set of customers to each other in the hope that this will save them. They hope to stick a single router between their two networks, and they have given Ben Bitdiddle, who is taking 6.829, the task of configuring the router. Ben decides to perform a few simulations studies in `ns` in order to understand better the details of configuring the router.

1. Ben begins his study by examining the dynamics of a simple TCP connection from AlphaWidgets to BetaGadgets over his router. Download Ben's script from `http://nms.lcs.mit.edu/6.829/ps/ps2/LFN.tcl` and examine it.
   (a) What is the round-trip time (RTT) and bandwidth of the links that Ben is simulating? How fast is his router? What kind of queueing discipline does his router use?
   (b) Run the script and plot the evolution of the sender's TCP window over time. Identify where TCP slow-start ends and where congestion-avoidance begins.
2. Ben looks at a map of BetaGadgets' network and notices that some of BetaGadgets' traffic travels over satellite links. He estimates that the average RTT for a connection between AlphaWidgets and BetaGadgets over satellite is 500 ms.
   (a) Reconfigure Ben's script so that the RTT for the connection is 500ms and repeat parts (a) and (b) above.[1] (Note: You will need to increase the length of the simulation in order to see TCP enter the steady state.)
   (b) Without looking at the simulation results, tell us how big the average congestion window would need be to allow a single TCP of fully utilizing the bottleneck link.
   (c) From your plot in 2-(a), what is the maximum value of TCP's congestion window size in the steady state? What is peculiar about the evolution of TCP's congestion window over time in this case (use part (a))?

[1] We want you to make the round trip propagation delay 500ms. You can't control the queuing delay. Anyway, the queuing delay in this setting is negligible.

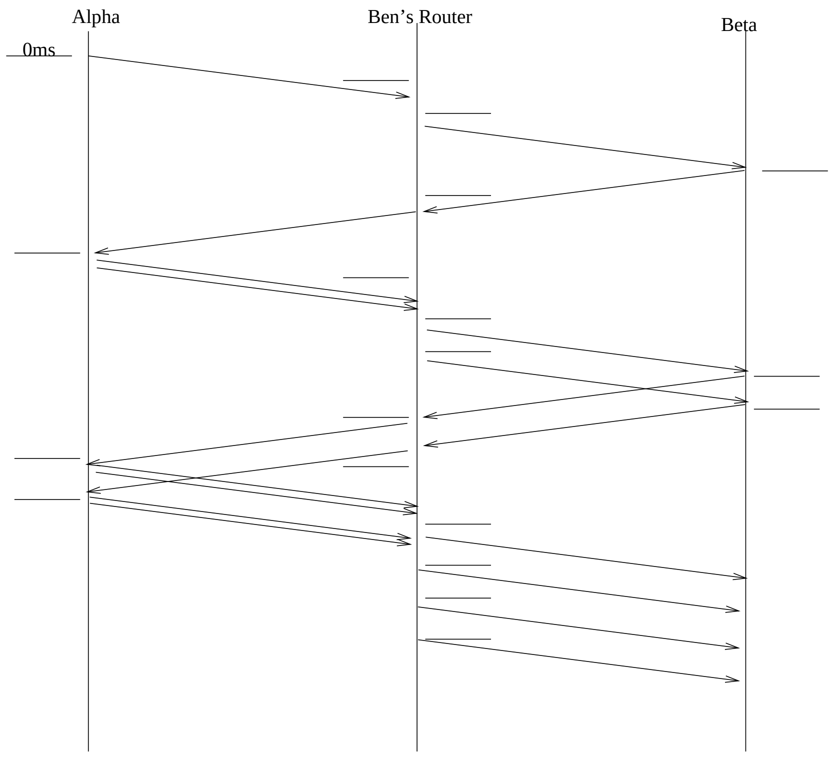

Figure 1: Timing diagram for TCP satellite connection from AlphaWidgets to BetaGadgets. Assume that it takes 0ms for the endpoints to process packets and for the router to process acknowledgments.

3. Ben figures that the problem with his satellite simulation has something to do with the size of the queue at the router; packets seem to be getting dropped as the TCP connection bursts traffic in slow start.

   (a) Plot the evolution of the queue over time as well along with the evolution of the sender's congestion window.

   To understand how the queue evolves over time, Ben sets up the timing diagram shown in Figure 1. In this figure, time proceeds vertically downwards, and the arrows indicate the progress of packets back-and-forth across the network.

   (b) Fill in the picture with the appropriate arrival and departure times for each packet. For the purposes of this exercise, you may assume that the packet takes 125ms to propagate through each leg of its trip, that packets take 0ms to process at the endpoints, and that the processing time for acknowledgments is negligible.

   (c) During slow-start, how is the maximum queue length related to the current congestion window size *cwnd*?

   (d) Assume you don't have control over the router to increase its maximum queue length. Can you change the behavior of the TCP sender so that it can achieve a large maximum cwnd and fully utilize the link? How?[2]

[2]Don't simulate your solution, just tell us what you would do.

2 Fun with RED and active queue management (AQM)

This problem is to help you understand different queue management schemes. This problem uses the file `AQM.tcl` in `http://nms.lcs.mit.edu/6.829/ps/ps2/`. You will also find the files in `http://nms.lcs.mit.edu/6.829/ps/ps2/scripts/` useful in data processing and trace analysis. (But feel free to write any code you want to!)

Ben just learned about several queue management schemes, including RED and RED with ECN, and Weighted Fair Queueing (WFQ) from 6.829. Ben wants to compare these schemes. Help Ben set up the simulation topology shown in Figure 2 in the file `AQM.tcl`. Let $BW(n_1, n_2)$ be the bandwidth of the duplex link between n_1 and n_2. Set $BW(S_i, R_0) = 100Mbps$, $BW(R_1, D_i) = 100Mbps$, $\forall i = 0, 1, 2$. Set $BW(R_0, R_1) = 1.5Mbps$.

1. Fair Queueing.

   (a) Ben first configures the R_0-R_1 link to use fair queueing. Let S_0 be a CBR source sending 1000-byte packets at a rate of 200 packets every second, and S_1 and S_2 be long-running TCP sources. Plot the goodput each source sees over time. Does it match your intuition? Now set $b_0 = BW(R_1, D_0) = 56Kbps$ (see `create-net1`). What goodput does each source see in this case? Why?

   (b) Now make S_0 a TCP source too. What goodput does each source see for the cases where $b_0 = 100Mbps$ and $b_0 = 56Kbps$? Why?

   (c) Based on this study, can you tell what the pros and cons of deploying fair queueing are?

2. RED.

   (a) Next, Ben sets a RED queue with parameters $minthresh\_ = 6$, $maxthresh\_ = 18$, $maxp\_ = 0.1$, $q\_weight\_ = 0.002$, $gentle\_ = true$ on the R_0-R_1 link. The buffer size on this link is set to be 54 packets (see `opt(qlen)`). All sources are long running TCPs. What are the goodput and loss rates of each connection?

3. RED & ECN.

   (a) Ben hears the hype that ECN is likely to be deployed in the next few years. So, he makes all the TCP sources ECN-capable. What does he see this time? Does the goodput of each source increase? What about the loss rate?

   (b) Ben wants to see how ECN-capable flows interact with non-ECN flows. So, he sets up S_0 to be ECN-capable and S_1, S_2 to be not. What does he see this time? Is there any benefit of deploying ECN in this network?

   (c) Ben becomes concerned about the possibility that an ECN-capable receiver might cheat to get higher throughput than it deserves. The form of cheating is simple: When the receiver gets a packet with ECN set on it, it *does not* echo it to the sender but instead suppresses this information.

   Ben sets up D_0 to be a cheating receiver; the others are honest ECN-capable connections. What are the goodput and loss rate of each source now? What happens when the number of cheating receivers varies?

4. Overall, how do you compare the fairness provided by the following queuing disciplines: FQ, RED, RED & ECN, and DropTail?

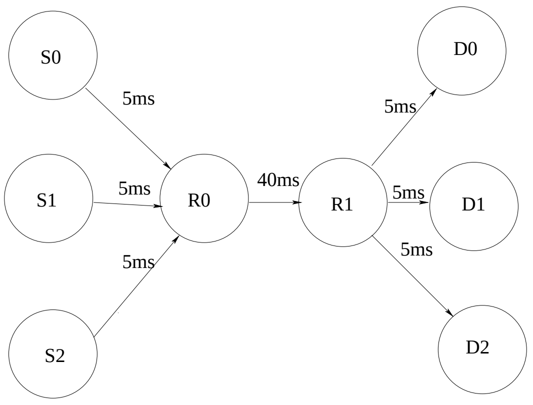

Figure 2: Simulation topology for the AQM problem.

3 How useful is traceback?

Suppose we had the ability to traceback IP packets deployed on network paths? Would this mitigate all attacks? Consider two cases: (i) Traceback is deployed on certain paths, and (ii) traceback is universally deployed.

Clearly describe what types of attacks can be handled (and how) using traceback, and what cannot. You might also speculate on additional helpful mechanisms that may be used in conjunction with traceback.

4 Buffering in a fast router

Louis Reasoner has been recruited by a hot new startup to design the packet queueing component of a high-speed router. The link speed of the router is 40 Gigabits/s, and he expects the average Internet round-trip time of connections through the router to be 100 ms. Louis remembers from his 6.829 days that he needs to put in some amount of buffering in the router to ensure high link utilization in the presence of the TCP sources in the network growing and shrinking their congestion windows. Louis hires you as a consultant to help design his router's buffers.

Assume for the purposes of these questions that you're dealing with exactly one TCP connection with RTT 100 ms (it turns out that this assumption does not change the results too much, for a drop-tail gateway, *if* the connections all end up getting synchronized). Also assume that the source is a long-running TCP connection implementing additive-increase (increase window size by 1/W on each acknowledgment, with every packet being acknowledged by the receiver, such that the window increases by 1 every RTT) and multiplicative-decrease (factor-of-two window reduction on congestion). Assume that no TCP timeouts occur. Don't worry about the effects during slow start (that was dealt with in Problem 1); this problem is about the effects on router buffering during TCP's AIMD congestion-avoidance phase.

You should (be able to) answer this question without running any simulation.

1. Show that if the amount of buffering in the router is equal to the product of bandwidth and round-trip delay (the *bandwidth-delay product*), the TCP connection can achieve a 100% link utilization (or very close to it). How much memory does this correspond to for the router under consideration?

2. Louis is shocked at how much memory is needed and thinks he may not be able to provide this much buffering in his router. He asks you what the average link utilization is likely to be when the amount of buffering in the router is very small compared to the bandwidth-delay product. Explain your answer.

3. Louis decides that this utilization will not do. He asks you how the average link utilization, U, varies as a function of r, the ratio of the amount of router buffering, B, to the "pipe-size," P (the bandwidth-delay product). Calculate $U(r)$ for $0 \leq r \leq 1$. (We know, from part 1, that $U(1) = 1$, and the answer to part 2 is $U(0)$.)

4. Sketch U versus r, for $0 \leq r \leq 1$. From this curve, what can you conclude about the gain in link utilization as you add more memory?

Hint: A good way to think about this problem is in terms of the TCP congestion window vs. time plots and look at how P, B, and the (time-varying) TCP congestion window size relate to each other. Think about the "steady-state" of a TCP connection and about how much data a TCP can send in one round-trip time. Don't worry about the TCP receiver flow control window limitation (i.e., assume that the receiver has a very large buffer).

5 Rate guarantees in a high-speed switch

Lem A. Prover has been asked to design a scheme for providing *rate guarantees* to queues in a crossbar-based CIOQ packet switch. She thinks about the problem a little and thinks that if she simply implemented weighted fair queueing on the queues, she might be able to solve the problem.

Recall that weighted fair queueing is a *relative* rate sharing scheme, while what Lem is looking for is an *absolute* rate guarantee scheme.

Formally, consider an $N \times N$ switch with N egress links in all and Q queues per egress link. The switch has speedup 2. Normalize all rates such that every ingress and egress link has rate 1. The switch is configured so that on a given egress link, queue i is guaranteed a rate g_i (with $0 \leq g_i \leq 1$ and $\sum_{i=1}^{Q} g_i \leq 1$).

The service model required for guaranteed service is as follows: *If queue i has an* arrival rate *of a_i and a guaranteed rate of g_i, then the queue's* service rate *is at least* $\min(a_i, g_i)$.

Assume that in each time-slot the switch does a maximal matching; with speedup 2, this means that *if the arrival rates on the ingress-egress pairs form a doubly substochastic matrix*,[3] the switch can support 100% throughput, i.e., no queue grows unbounded at an ingress and all packets that arrive get transmitted to the appropriate egress.

1. Assume that the arrival rates on the ingress-egress pairs form a doubly substochastic matrix. Prove that there is a choice of weights in a weighted fair queueing system that will allow Lem to implement the rate guarantee service model described above.

2. Give an example of arrival rates that are *not* doubly substochastic, where the weighted fair queueing approach of the previous part *cannot* meet the rate guarantee service model.

3. Lem wants to see if she can prove the result we mentioned in lecture 6, that if the arrival rates to ingress-egress pairs form a doubly substochastic matrix, then a maximal matching scheduling scheme in each time-slot running on a switch of speedup 2 provides 100% throughput.

   She decides to consider a simplified version of this problem. Suppose that in each time-slot *at most* one packet arrives on an ingress link and *at most* one packet arrives *for* any given egress link. (I.e., the arrival rates are doubly substochastic even when measured over a single time-slot duration.) Let q_{ij} be the current queue length of packets at ingress i destined for egress j. Prove that if the switch does a maximal matching in each time-slot and has speedup $S \geq 2$, then each q_{ij} has a finite upper-bound.

   Hint: Look at the sum of all the queue lengths, $\sum_{i,j} q_{ij}$ and consider the increment and decrement in this quantity in each time-slot. If you show that the sum of all queue lengths is bounded, then you're done because each queue length is ≥ 0.

[3] In a doubly substochastic matrix, the sum of each row is ≤ 1 and the sum of each column is ≤ 1.

6 XCP with lying participants

After reading the XCP paper in 6.829, Alyssa P. Hacker and Cy D. Fect get into an argument about how fragile XCP is in the presence of senders who lie about the information they are given. Consider a version of XCP where the sender reports to the network its RTT and its throughput; i.e., the fields in the congestion header are: (i) RTT (ii) throughput (instead of cwnd), and (iii) feedback. This change will make it easier for you to reason about this problem.

The assumption we make is that the sender can lie about the RTT and throughput to the network, in an attempt to persuade the network to give it more (or less!) than its correct share, or in an attempt to mess up the resulting link utilization (causing the network to run under-utilized or in persistent congestion). However, the sender does not *send* at an arbitrary rate; it always updates its congestion window according to the feedback the network sends. More precisely,

$$cwnd \leftarrow cwnd + feedback \cdot \frac{true\_RTT}{declared\_RTT}.$$

This adjustment ensures that the increase in the sender's throughput is the one intended by the network.

Alyssa and Cy get into an argument about what happens when users misbehave and lie to the routers. Help them resolve their argument. Be precise and concise in your answers to these questions. (These questions are intentionally somewhat under-specified, to encourage you to think about all the things that can go wrong.)

1. Suppose the sender(s) lie only about RTT. What would XCP's performance in terms of link utilization and fairness be? (A sender might lie in either direction, of course.)
2. Suppose the sender(s) lie only about throughput. What would XCP's performance in terms of link utilization and fairness be? (A sender might lie in either direction, of course.)
3. Suppose you were to design a method to detect misbehaving senders in an XCP network by placing monitoring agents at various points in the network. Describe the key features of your design, including where the agents ought to be placed and how they would detect misbehaving sources. Discuss the following aspects of your solution: (i) where the agents will be located, (ii) what do the agents do, (iii) how much state they maintain, (iv) the complexity of your solution.
4. What would you do to the packets of senders that are detected to be liars by your method? Explain your answer.