

Internetworking and End-to-End Protocols

Assigned reading: Peterson and Davie: Chapters 4, 5 and 9.1. All problems carry equal weight. For full credit, show all work.

1. TCP RTT Estimation

One difficulty with the original TCP SRTT estimator is the choice of an initial value. In the absence of any special knowledge of network conditions, the typical approach is to pick an arbitrary value, such as 3 seconds, and hope this will converge quickly to an accurate value. If this estimate is too small, TCP will perform unnecessary retransmissions. If it is too large, TCP will wait a long time before retransmitting if the first segment is lost. Also, the convergence might be slow.

- a. Choose $\alpha = 0.85$ and $SRTT(0) = 3$ seconds, and assume all measured RTT values = 1 second with no packet loss. What is $SRTT(19)$? Recall, $SRTT(k + 1) = \alpha * SRTT(k) + (1 - \alpha) * RTT(k + 1)$.
- b. Now let $SRTT(0) = 1$ second and assume RTT values = 3 seconds and no packet loss. What is $SRTT(19)$?
- c. Now consider $\alpha = 0.95$. Repeat parts a. and b. and comment on the effect of a larger or smaller α .

2. TCP Slow Start

Although slow start with congestion avoidance is an effective technique for coping with congestion, it can result in long recovery times in high-speed networks.

- a. Assume a roundtrip time delay of 125 ms (about what might occur across a continent) and a link with an available bandwidth of 1.2 Gbps and a segment size of 512 octets. Determine the window size needed to keep the pipe full and the time it will take to reach that window size after a time out using the Jacobson Algorithm.
- b. Repeat for a segment size of 32 Kbytes.

3. TCP Congestion Performance

Consider a TCP system implementing slow start and congestion avoidance with fast retransmit and fast recovery. When a connection is setup the congestion window is initialized to one segment and the slow start threshold to 64 segments. To simplify the problem, assume that the timeout is equal to the RTT (an exact estimate) and specify time in units of RTT, such that one time slot is one RTT.

Packet transmissions are such that at each time slot the sender sends all packets in the congestion window. If ACK's are received in the next time slot, there is no timeout. In addition, to simplify matters either the entire window is acknowledged or none of its segments are acknowledged.

For a particular connection, ACK's are received in time slots 1-7, 9-30, 32-40, and 42-50. Timeouts occur in slots 8 and 31; in slot 42, three duplicate ACK's are received for the packets sent in time slot 41.

For the system described, plot both the congestion window and the slow start threshold (on the same graph) versus time (slots). Remember to consider the differences between slow start and congestion avoidance with fast retransmit and fast recovery when changing the congestion window.

4. Adaptive Retransmission, Part I

- a. What is the retransmission ambiguity problem addressed by the Karn-Partridge algorithm? How does the algorithm avoid the ambiguity?
- b. Write a C program to estimate the frequency of unnecessary retransmission with the original TCP adaptive retransmission algorithm. In particular, write a one-million-iteration loop that calls a function

```
double get_rtt ();
```

- to measure RTT, calculates an estimated RTT, and counts the number of times that the next measured RTT exceeds the current estimate. Use a damping factor of 0.15 (α on P&D p. 404 is between 0.8 and 0.9) and double-precision floating-point variables. Programs longer than 50 lines will receive no credit.
- c. Repeat part (b) for Jacobson's algorithm (Jacobson-Karels in P&D, defined on p. 405-406), using $\delta = 0.15$.

5. Adaptive Retransmission, Part II

Execute your code from problem 4 parts (b) and (c) using the `get_rtt()` function given below and report the frequency of unnecessary retransmissions as a percentage of packets (rounded off to an integer).

6. Fair Queueing

A router implements fair queueing of four incoming flows over a single outgoing channel. In this problem, you will determine the order of packets sent out from the router and calculate statistics from your results. Assume for simplicity that the problem starts at time $t = 0$ and that sending a packet of length N requires N time units. Packets arrive on the four flows, named A, B, C, and D, as follows:

- A packets of length 10 arrive at times 0, 5, 10, 15
- B packets of length 8 arrive at 25, 45, 65, 85, 105
- C packets of length 5 arrive at 10, 20, 30, 40, 50, 60, 70, 80
- D packets of length 4 arrive at 1, 2, 30, 31, 32, 33, 80, 81, 82, 83

All service counters start at 0 at time 0.

- a. For each packet sent on the outgoing link, record the start time, the service counter for each flow at that time, and the name of the packet (*e.g.*, write *B3* for flow B's third packet).
- b. Calculate the percentage of the link used by each flow up to time 100. Was the link fairly distributed? Why or why not?
- c. Calculate the average packet delay for each flow—the average difference between the arrival time of the packets and the time that they begin to be sent on the outgoing link. Explain the differences in the averages in terms of the burstiness of the arrivals.
- d. Repeat parts (a), (b), and (c) using round-robin scheduling. Skip queues that are empty on their turn. Explain the differences in utilization and delay in comparison with fair queueing.
 - a.

7. nslookup Network Utility

Use the `ews` machines for this problem. Show the commands that you use to solve the problem and the output you get. No credit if you don't show your work. The `nslookup` utility allows you to query domain name servers for the internet. Review the section of the text on domain name servers and read the man page for `nslookup`. You can enter the interactive mode of `nslookup` by simply typing `nslookup`, and from there you can type `?` for a list of commands.

- a. Find the names of the nameservers known to `cs.uiuc.edu`.
- b. Find the canonical name and IP address for the machine with alias `www.cs.uiuc.edu`.
- c. Find the names of the mailserver(s) for the machine with alias `www.cs.uiuc.edu`