Homework Assignment #4

Due: Friday Dec. 7th @ 3:50PM

EE122: Introduction to Communication Networks

(Fall 2007)

Department of Electrical Engineering and Computer Sciences
College of Engineering
University of California, Berkeley

Vern Paxson / Jorge Ortiz / Lisa Fowler / Daniel Killebrew

Turn in as **hardcopy** to the drop box in **240 Cory**.

1. (a) Kurose & Ross, Chapter 2, p. 186:
    i. P21.
        A. The advantage of sending the QueryHit message directly over a TCP connection from Bob to Alice is that the QueryHit message is routed by the underlying Internet without passing through intermediate peers; thus, the delay in sending the message from Bob to Alice should be substantially less. The disadvantage is that each peer that has a match would ask Alice to open a TCP connection; Alice may therefore have to open tens or hundreds of TCP connections for a given query. Furthermore, there will be additional complications if Alice is behind a NAT, and thus can't accept incoming connection requests.
        B. When a Query message enters a peer, the peer records in a table the MessageID along with an identifier of the TCP socket from which the message arrived. When the same peer receives a QueryHit message with the same MessageID, it indexes the table and determines the socket to which it should forward the message.
        C. When the Query message reaches Bob, it contains an ordered list of all the IP addresses of the peers the message passed through between Alice and Bob. When Bob sends back a QueryHit message, Bob includes a copy of the ordered list in the message. When a peer receives the QueryHit message, it can use the list to determine the next peer in the reverse path.
    ii. P22.
        A. Each super-duper peer is responsible for roughly 200^2 = 40,000 nodes. Therefore, we would need about 100 super-duper peers to support 4 million nodes.

B. Each super peer might store the meta-data for all of the files its children are sharing. A super-duper peer might store all of the meta-data that its super-peer children store. An ordinary node would first send a query to its super peer. The super peer would respond with matches and then possibly forward the message to its super-duper peer. The super-duper peer would respond (through the overlay network) with its matches. The super-duper peer may further forward the query to other super-duper peers.

(b) Kurose & Ross, Chapter 6, pp. 581-582:

i. P8, parts (a), (b), (c), and (e).

(a) Messages from C to A need to be relayed by B, since A can't hear C. This means that one slot is spent transferring a message from C to B, and another from B to A, so the answer is **1 message per 2 slots**.

(b) The transmissions do not interfere with one another since only B can hear A and only C can hear D, so the answer is **2 messages per slot**.

(c) Since B can hear both A and C, only one of them can transmit during any given slot, so the answer is **1 message per slot**.

(e)A. It takes two slots to get a message from C to A and another two to return the ACK from A to C, so **1 message per 4 slots**.

B. A can send to B concurrent with D sending to C, but the acknowledgments in the other direction will interfere.
So we have:
slot 1 Messages from A $\rightarrow$ B, D $\rightarrow$ C
slot 2 ACK from B $\rightarrow$ A
slot 3 ACK from C $\rightarrow$ D
(clearly the last two slots could come in the other order)
This gives us **2 messages per 3 slots**.

C. We have:
slot 1 Message from C $\rightarrow$ D
slot 2 ACK from D $\rightarrow$ C, message from A $\rightarrow$ B
slot 3 ACK from B $\rightarrow$ A
This again gives us **2 messages per 3 slots**.

(c) Kurose & Ross, Chapter 7, pp. 672-673:

i. P23.

A.

Time Slot	Packets in the queue	Number of tokens in bucket
0	1, 2, 3	2
1	3, 4	1
2	4, 5	1
3	5, 6	1
4	6	1
5	–	1
6	7, 8	2
7	9, 10	1
8	10	1

B.

Time Slot	Packets in output bucket
0	1, 2
1	3
2	4
3	5
4	6
5	–
6	7, 8
7	9
8	10

(d) Kurose & Ross, Chapter 8, p. 751:

i. P9.

The idea behind this sort of third-party setup (which by the way is how the fairly popular "Kerberos" system works) is that the KDC mediates communication between Alice and Bob because both Alice and Bob trust the KDC (to the extent of sharing their keys with it). A corresponding exchange could look like:

A. Alice sends a request to the KDC stating her identity, A, and that she wishes to communicate with Bob. She encrypts this using K_{A-KDC}, which keeps the message private between Alice and the KDC, and authenticates Alice to the KDC.

B. The KDC generates a random session key K_S and encrypts it *twice*, as follows.
First, the KDC encrypts both A and K_S using K_{B-KDC}, resulting in:

$$E(\langle A, K_S \rangle, K_{B-KDC}) \tag{1}$$

a glob of bits that Alice can't read, but from which Bob will be able to extract both the session key and the fact that the KDC issued the key to Alice.
Second, the KDC encrypts both this ciphertext and K_S (again) but this time using K_{A-KDC}, i.e., generating:

$$E(\langle K_S, E(\langle A, K_S \rangle, K_{B-KDC}) \rangle, K_{A-KDC}) \tag{2}$$

and sends this to Alice.

C. Alice recovers K_S by applying K_{A-KDC} to the message given in (2). She also recovers (1) from the KDC and sends this to Bob.
Bob can decrypt (1) in order to recover both K_S and the identity A, which lets Bob know that the KDC intended K_S for use between Bob and Alice.

2. **Sizing Header Fields**. You are designing a reliable, sliding window, byte-stream protocol similar to TCP. It will be used for communication with a geosynchronous satellite network, for which the bandwidth is 1 Gbps and the RTT is 275 ms. Assume the maximum segment lifetime is 30 seconds.

(a) How many bits wide should you make the *AdvertisedWindow* and *SequenceNum* fields?
Answer. To fully utilize the network, *AdvertisedWindow* needs to be larger than $(\textit{Delay} \times \textit{Bandwidth})$.

$$\begin{aligned}(\textit{AdvertisedWindow}) &\geq (\textit{Delay}) \times (\textit{Bandwidth}) \\ &= 275\textit{ms} \times 1\textit{Gbps} \\ &= 275\textit{Mbit} \\ &= 34.375\textit{MB}\end{aligned}$$

$$\begin{aligned}2^{25} &= 33,554,432 \\ 2^{26} &= 67,108,864\end{aligned}$$

Therefore, 26 bits are needed for $AdvertisedWindow$.

SequenceNum needs be large enough that the sequence number does not wrap around before any delayed segments have left the network, which is presumed to occur within the maximum segment lifetime.

$$\begin{aligned}(\textit{SequenceNum}) &\geq (\textit{Maximum Segment Lifetime}) \times (\textit{Bandwidth}) \\ &= 30\textit{s} \times 1\textit{Gbps} \\ &= 30\textit{Gbit} = 3.75\textit{GB}\end{aligned}$$

$$\begin{aligned}2^{31} &= 2,147,483,648 \\ 2^{32} &= 4,294,967,296\end{aligned}$$

Therefore, 32 bits are needed for *SequenceNum*.

(b) If *AdvertisedWindow* is 16 bits, what upper bound would that impose on the effective bandwidth?

Answer. If *AdvertisedWindow* is 16 bits, it is smaller than $(\textit{Delay} \times \textit{Bandwidth})$ product. Therefore, *AdvertisedWindow* is the limiting factor of the effective bandwidth. During a single *RTT*, only *AdvertisedWindow* amount of data can be transferred.

$$\begin{aligned}(\textit{Effective Bandwidth}) &= \frac{(\textit{AdvertisedWindow})}{\textit{RTT}} \\ &= \frac{2^{16}\textit{B}}{275\textit{ms}} \\ &= 238.31\textit{KB/s}\end{aligned}$$

(c) If it turns out that 0.5% of the packets sent over the path are lost, what throughput would you expect a long-running TCP connection to achieve?

Assume a value of *AdvertisedWindow* large enough to not impede performance.

Answer. The average long-term performance achieved by a TCP sender is governed by the *TCP Throughput Equation*:

$$\begin{aligned}(\textit{Throughput}) &= \frac{\sqrt{1.5}B}{RTT\sqrt{p}} \\ &= \frac{\sqrt{1.5}B}{275ms\sqrt{0.005}} \\ &= (62.98 \times B) \text{ bytes/s}\end{aligned}$$

Assuming $B = 1500$ bytes:

$$\begin{aligned}(\textit{Throughput}) &= (62.98 \times B) \text{ bytes/s} \\ &= 94{,}475 \text{ bytes/s}\end{aligned}$$

3. **QoS.**

(a) Suppose the capacity C of a link is 18. Assume that 4 sources—S1, S2, S3, and S4—are trying to send over the link at rates of $r_1 = 2$, $r_2 = 4$, $r_3 = 5$, and $r_4 = 8$, respectively. What is the max-min fairness allocation?

Answer. In the first round, we have $N = 4$ sources needing allocations, so the fair share is $18/4 = 4.5$. This suffices for sources 1 and 2 (giving them their full requested allocation of 2 and 4, respectively), so we remove them and repeat the process.

For the second round, $N' = 2$ and we have already given out $2 + 4 = 6$ of the total capacity of 18. Therefore, 12 remains, and the fair share is $12/2 = 6$. This suffices for source 3, so it gets its full requested allocation of 5. This leaves us with a remaining capacity of 7, with $N'' = 1$. The fair share is 7. All remaining sources (just source 4) want more than that, so each is allocated the remaining fair share.

Thus, the allocations are: A1=2, A2=4, A3=5, A4=7.

(b) For each of the following statements, indicate whether it applies to Integrated Services (IntServ), Differentiated Services (DiffServ), and/or Best Effort. (A given statement can apply to more than just one type of service.)

i. The service is provided end-to-end.
Answer. IntServ and Best Effort. DiffServ operates only between domains and not end-to-end, while IntServ and Best Effort both are provided as end-to-end services.
Note: it's easy to not think of Best Effort as providing any actual sort of service, and therefore not considering that it provides a service end-to-end.

ii. Among the three, requires the most state in routers.
Answer. IntServ. IntServ requires the most state, since it needs to track individual flows or connections. DiffServ only needs to maintain per-class state, of which there are not many classes. Best Effort doesn't maintain any state.

iii. Is widely available in the Internet today.
Answer. Best Effort (with DiffServ also being allowed in addition).
Best Effort is the only end-to-end service widely available today. We discussed how DiffServ is frequently available within individual domains, though usually not between domains. Because the question wasn't clear on just what constitutes "widely available," answers that included DiffServ too were allowed.

iv. Provides isolation and guarantees among aggregated flows but not individual connections.
Answer. DiffServ. DiffServ operates on large aggregates. IntServ provides fine-grained isolation and guarantees (which makes it more difficult to deploy, since it requires more state). Best Effort doesn't provide any isolation or guarantees, period.

4. **Queueing Theory.** Jorge wants to find the average number of people in his office hours, which are from 1PM-2PM. He observes the following three people and the times that they arrive and leave:

   - Alice: 1:00-1:20PM
   - Bob: 1:10-1:45PM
   - Eve: 1:40-2PM

   Use Little's Law to compute the mean number of people in his office.

   Answer. Let λ be the mean arrival rate of people, and let d be the mean time people spend in the office. Let N be the mean number of people in the office. Little's Law states:

$$N = \lambda d$$

   We have $\lambda = 3$ people/hour. We can compute:

$$\begin{aligned} d &= \frac{20 \text{ minutes} + 35 \text{ minutes} + 20 \text{ minutes}}{3 \text{ people}} \\ &= \frac{75 \text{ minutes}}{3 \text{ people}} \\ &= 25 \text{ minutes/person} \end{aligned}$$

   Applying Little's Law:

$$\begin{aligned} N &= \lambda d \\ &= 3 \text{ people/hour} \cdot 25 \text{ minutes/person} \\ &= 3 \times \frac{25}{60} \\ &= 1.25 \end{aligned}$$

   Thus, the mean number of people in Jorge's office is 1.25.

5. **Time-Sequence Plots.**

   (a) For the packet trace (recorded using `tcpdump`) at `http://inst.eecs.berkeley.edu/%7Eee122/fa07/hw/hw4-trace1.tcpdump`, construct a time-sequence plot for the data packets . . .

   See figures 1–3 for the time-sequence plots.

   . . . and acknowledgments, and use it to identify:

   - The approximate RTT.
     Answer: 280 ms
   - The overall throughput.
     Answer: 36 KB/s
   - The largest effective window size used.
     Answer: 33 KB. Note, this refers to the maximum amount of data in flight (which here occurs near the end of the trace), *not* the advertised window (unless it happens to limit the maximum amount of data in flight—not the case for any of these traces).
   - The approximate maximum throughput obtained over two or more flights of packets.
     Answer: $\approx$ 90 KB/s (last two flights). Note, there are a lot of ways to measure throughput (data in flight, data until acknowledged, throughput as echoed in acknowledgments due to "ack clocking"), so we allowed any answer here within a factor of two of our best answer.
   - Whether the connection is ever limited by the advertised window.
     Answer: the advertised window was never a limit.

   (b) Construct time-sequence plots for the packet traces at `http://inst.eecs.berkeley.edu/%7Eee122/fa07/hw/hw4-trace2.tcpdump` and `http://inst.eecs.berkeley.edu/%7Eee122/fa07/hw/hw4-trace3.tcpdump`. For these traces, identify retransmitted packets and for each note whether it occurred due to Timeout or Fast Retransmission. For the latter, does the plot indicate that the sender also used Fast Recovery?

   Answer: for the first of these traces (`hw4-trace2.tcpdump`), packets 24001 and 72501 were retransmitted due to Fast Retransmission. We can tell that Fast Recovery is also used because as more duplicate ACKs arrive, additional data is sent.

   For the second trace, packets 4097, 28673, 35841 and 58881 are retransmitted due to timeout.

Include copies of your plots with your answers.

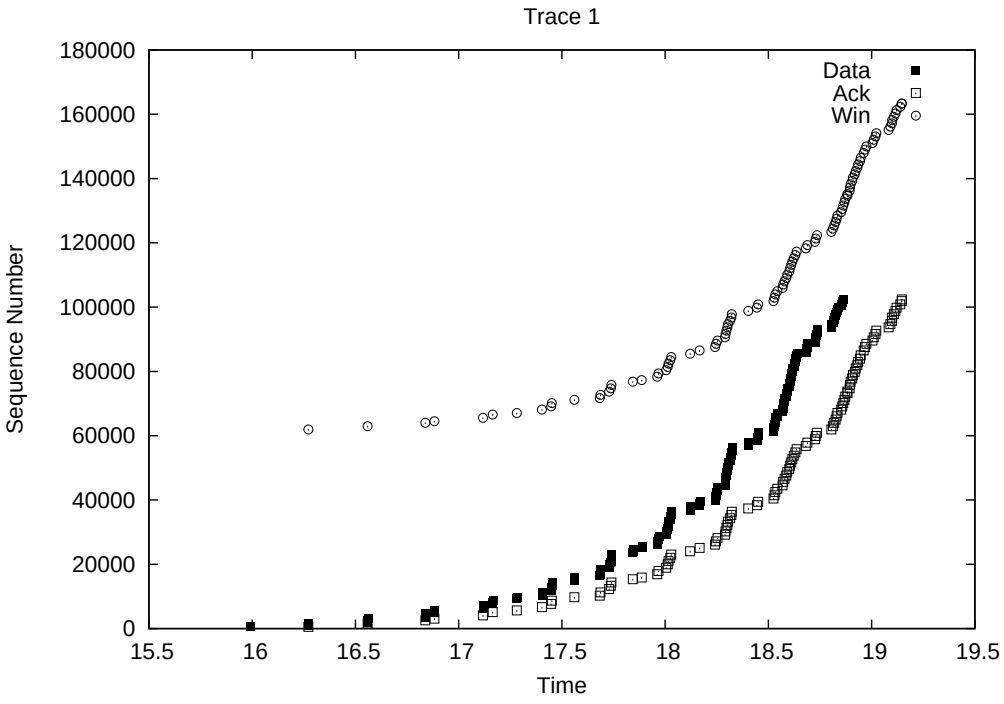

Figure 1: Time-Sequence Plot for `hw4-trace1.tcpdump`.

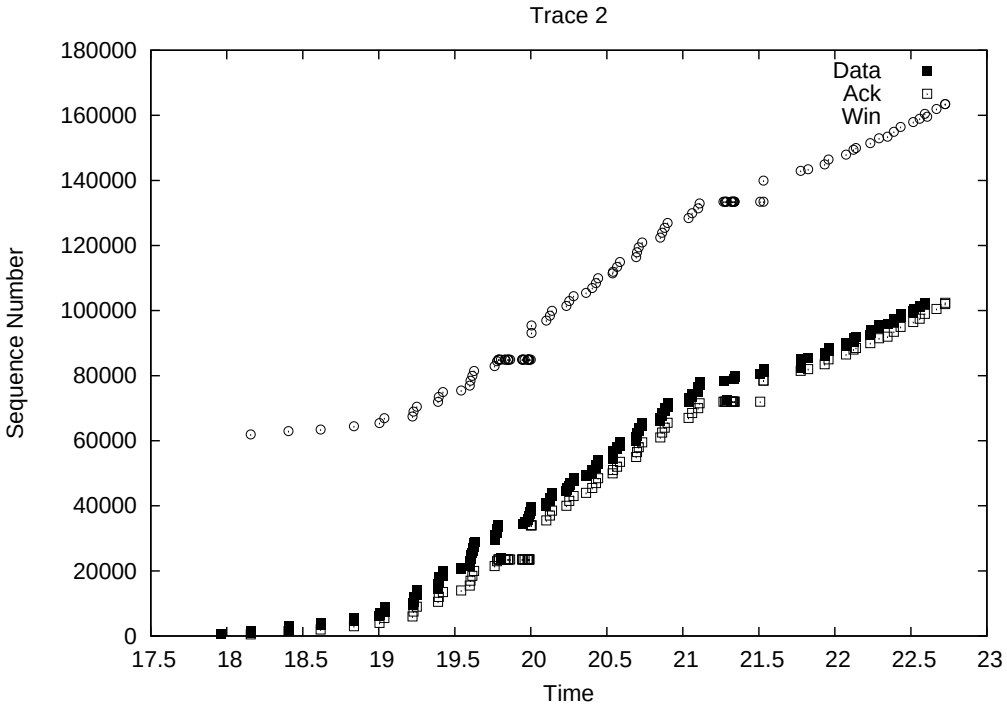

Figure 2: Time-Sequence Plot for `hw4-trace2.tcpdump`.

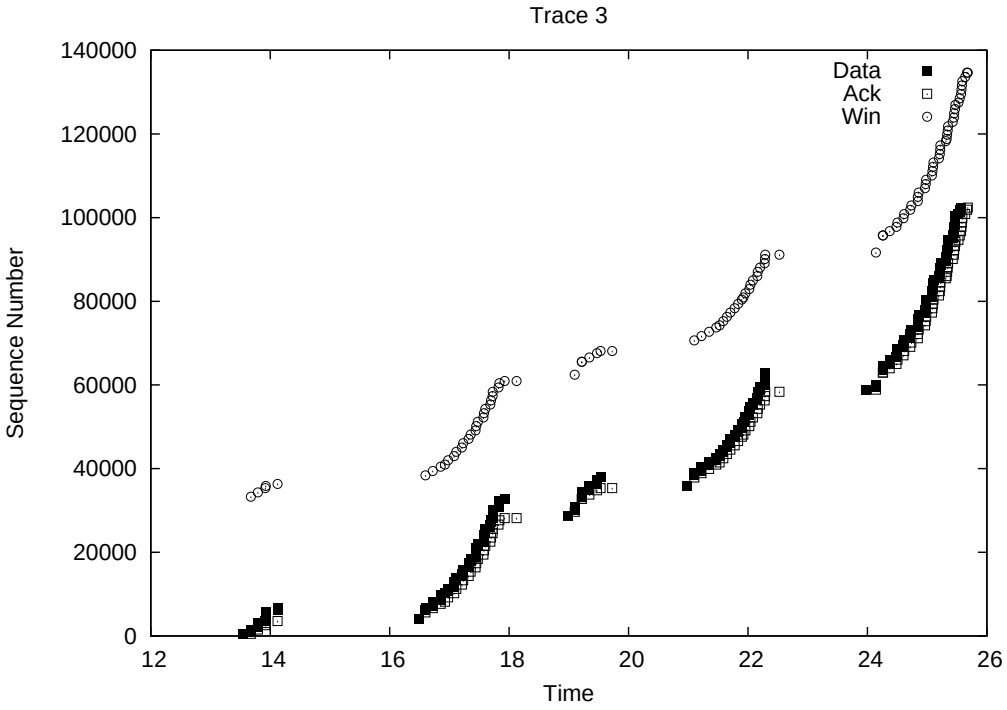

Figure 3: Time-Sequence Plot for `hw4-trace3.tcpdump`.