ipp05 Assignment 6
Assigned 9/29/05
Due 10/7/05

email answers to dunigan@cs.utk.edu

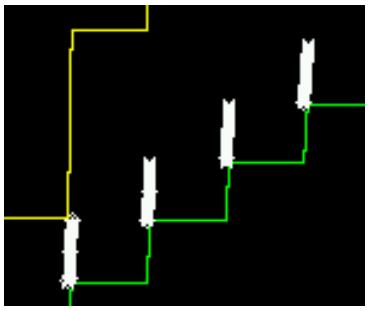
Points 35

Objective: gray matter stimulation

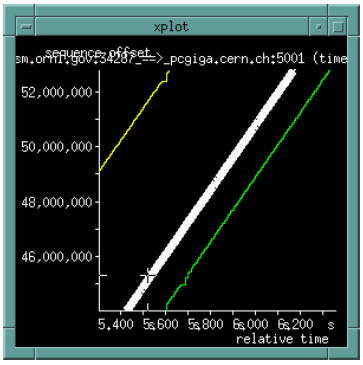
1. Suppose you have a network path between hosts A and B that has a one-way delay of 60ms. Suppose both A and B have network interfaces (NIC) with max speeds of 200,000 bits/second. How long does it take the last bit of a 1000 byte packet sent out from Host A to be read by Host B?
2. What is the purpose of the readn() function described in class?
3. Consider a path between hosts A and B with a RTT of 100 ms. Starting with one segment (cwnd=1),
   - **(a)** how many RTTs will it take for slow-start to reach a window size of 32 segments. (assume no delayed ACKs)
   - **(b)** How long in seconds?
   - **(c)** How long will it take if we start with 4 segments initially (cwnd=4)?
4. TCP Sender has transmitted packets 51 thru 58. At the receiver the packets arrive in the following order: 51 52 54 55 56 57 53 58
   - (a) Show the ACK numbers that the receiving TCP sends back as these packets arrive.
   - **(b)** Describe the action a TCP Tahoe sender would take when receiving these ACKs
5. TCP for slow-starts sets cwnd=1 (segment) then doubles cwnd for each ACK received. For congestions avoidance, TCP sets cwnd = cwnd + 1/cwnd for each ACK received. Consider a "virtual MSS" of k segments (k=1 for standard TCP). Now at slow-start, we set cwnd=k (segments) and still double cwnd for each ACK.
   - (a) What is the effect on the performance (speed) of slow-start when using k segments?
   - (b) For congestion avoidance with our virtual MSS, we set cwnd=cwnd+k/cwnd for each ACK received. How does this affect performance (recovery time)?
   - (c) A GigE jumbo frame is 9000 bytes, compared to standard Ethernet frame size of 1500. Suppose k=6 in our virtual MSS scheme. How does slow-start and congestion avoidance for our virtual MSS compare with using jumbo frames?

6. Below are snapshots of a tcpdump/tcptrace. Which figure shows a flow that is receive buffer limited (a b or c)? Which shows a flow that is send buffer limited (a b or c)?

(a)



(b)



©

7. In our simple UDP client/server homework assignment, the server sends back a short ACK packet for each data packet received. Suppose we change the client to send out two 1460-byte packets back to back. We then issue a recvfrom(), and when the first ACK arrives we record the time (microsecond precision), and issue another recvfrom(). When the 2nd ACK arrives, we record the time when the 2nd recvfrom() completes. The difference of these two time values is the packet gap time.
    a. What might we be able to conclude about the bottleneck link speed from this gap time?
    b. Suppose the gap time is 120 ms, what is the link speed?
    c. Suppose the gap time is 24 ms, what is the link speed?
    d. Suppose the gap time is 60 ms, what is the link speed?

8. NetDude approaches the HELP desk with the following server program. NetDude says his professor has a program that will make a TCP connection to NetDude's server on port 17800 and send it 50,000 ASCII bytes. NetDude says his server program compiles just fine under cc (though gcc warns him about something in the "accept" statement, but NetDude is sure it's OK). His server program runs without any error messages, but just prints

Msg:

and he doesn't seem to be getting the data from the professor's client program. NetDude suspects the professor's client program is buggy. Can you help NetDude with his server code? (identify as many things as you can that are wrong with this code). (The include's are OK.) Line numbers at end are to be used in your answer.

```
#include        <stdio.h>
#include        <sys/types.h>
#include        <sys/socket.h>
#include        <netinet/in.h>
#include        <arpa/inet.h>
#include        <netdb.h>
char buff[50000];

main(argc, argv)
int     argc;
char    *argv[];
{
        int sockfd, newsockfd, clilen, servlen;    /* line 1 */
        struct sockaddr_in cli_addr, serv_addr;    /* line 2 */

        sockfd = socket(AF_INET, SOCK_DGRAM, 0);    /* line 3 */

        bzero((char *) &serv_addr, sizeof(serv_addr));  /* 4 */
        serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);  /* 5 */
        serv_addr.sin_port = 17800;                     /* 6 */

        bind(sockfd, (struct sockaddr *) &serv_addr, 0);  /*7*/
        newsockfd = accept(sockfd, cli_addr, clilen);    /* 8 */
        read(newsockfd,buff,sizeof(buff));              /* 9 */
        printf("Msg: %s\n",buff);
}
```

email answers to dunigan@cs.utk.edu