COMP/ELEC 429/556 Fall 2017
Homework #1

Assigned 9/28/2017
Due 10/12/2017 11:55pm
Submit Electronically to Canvas
(Hard deadline, no slip day may be used)
This homework is worth 10% of your final grade

IMPORTANT: To be completed by each student individually. See syllabus for policy details. Type set your responses either in the space provided in this document or in a separate document. No hand-written responses will be accepted.

Name: Xing Liu

Email: xl76@rice.edu

Student ID: S01276729

Notations: K=1,000, M=1,000,000, G=1,000,000,000, 1 byte (B) = 8 bits (b).

Problem 1 (20 points) – A student wants to measure as accurately as possible the time it takes to send 1GB, in 10KB chunks, to a remote computer using a `SOCK_STREAM` socket. To do so, he writes a sender program which contains the following code fragment:

```
/* some standard code to create a socket
 * and to establish a connection
 */
... code omitted ...

fcntl(send_socket, F_SETFL, O_NONBLOCK);
gettimeofday(&start, NULL);
for (i=0; i<100000; i++) {
        chunk = malloc(10000);
        send(send_socket, chunk, 10000, 0);
        free(chunk);
}
gettimeofday(&end, NULL);
/* compare start and end to obtain
 * the elapsed time
 */
```

There are two fundamental design flaws in this code fragment. One causes the program to sometimes send fewer than 1GB. Another one causes the program to be very inaccurate. Identify these flaws and provide a revised code fragment that removes these flaws. You may find the `man` pages on CLEAR for the various system calls useful.

My Answer:

The send() function may send less than 10KB sometimes, which cause the program send fewer than 1GB; and the program shouldn't malloc and free the chunk every time, it should do it just once.

Revised code:

```
fcntl(send_socket, F_SETFL, O_NONBLOCK);
gettimeofday(&start, NULL);
chunk = malloc(10000);
int count = 0;
int remain = 0;
for (i=0; i<100000; i++) {
   remain = 10000;
   while(remain != 0) {
      count = send(send_socket, chunk+count, remain, 0);
      remain -= count;
   }
}
free(chunk);
gettimeofday(&end, NULL);
```

Problem 2 (10 points) The weakness of the NRZI bit representation scheme is that a long sequence of 0's in the input bit stream will cause the electrical signal to contain no voltage transition for a long period of time, thus making it difficult for the receiver to detect clock drifts. One solution to this problem is to translate the input bit stream into an encoded bit stream that eliminates long sequences of 0's.

Design a way to translate every 3-bit sequence in the input bit stream to a 4-bit encoded sequence such that the resulting encoded bit stream is guaranteed to contain no more than 2 consecutive 0's. Write down your answer in the shaded boxes below.

3-bit input sequence		**4-bit encoded sequence**
000	→	0101
001	→	1011
010	→	0110
011	→	0111
100	→	1010
101	→	1101
110	→	1110
111	→	1111

Problem 3 (29 points) -- Suppose you are designing a multi-access (i.e. broadcast) wired network using CSMA/CD (Carrier Sense Multiple Access/Collision Detect) mechanisms similar to the Ethernet for media access control. Assume that in your design, the network's link speed is 15Mbps, frame sizes may range from 200 bytes to 2000 bytes. Also assume that data signal propagates on the network link at a speed of $2*10^8$ meters per second.

(a) To guarantee that a collision is detected, what is the maximum allowable link length (in meters) between any pair of hosts connected to the same link? Show your work. (15 points)

My Answer:

Network Length <= (min_packet_size)*(propagation_speed)/(2*bandwidth)
= (8*200b)*(2*10^8mps)/(2*15*10^6bps) ≈ 10667m

(b) State the reasons why broadcast Ethernet, where all hosts on the network share one single channel (i.e. wire) and CSMA/CD (carrier sense multiple access/collision detect) is used to arbitrate media access among the hosts, cannot (1) support a large number of hosts and still maintain high throughput, or (2) support hosts spread across a large geographic area. (14 points)

My Answer:

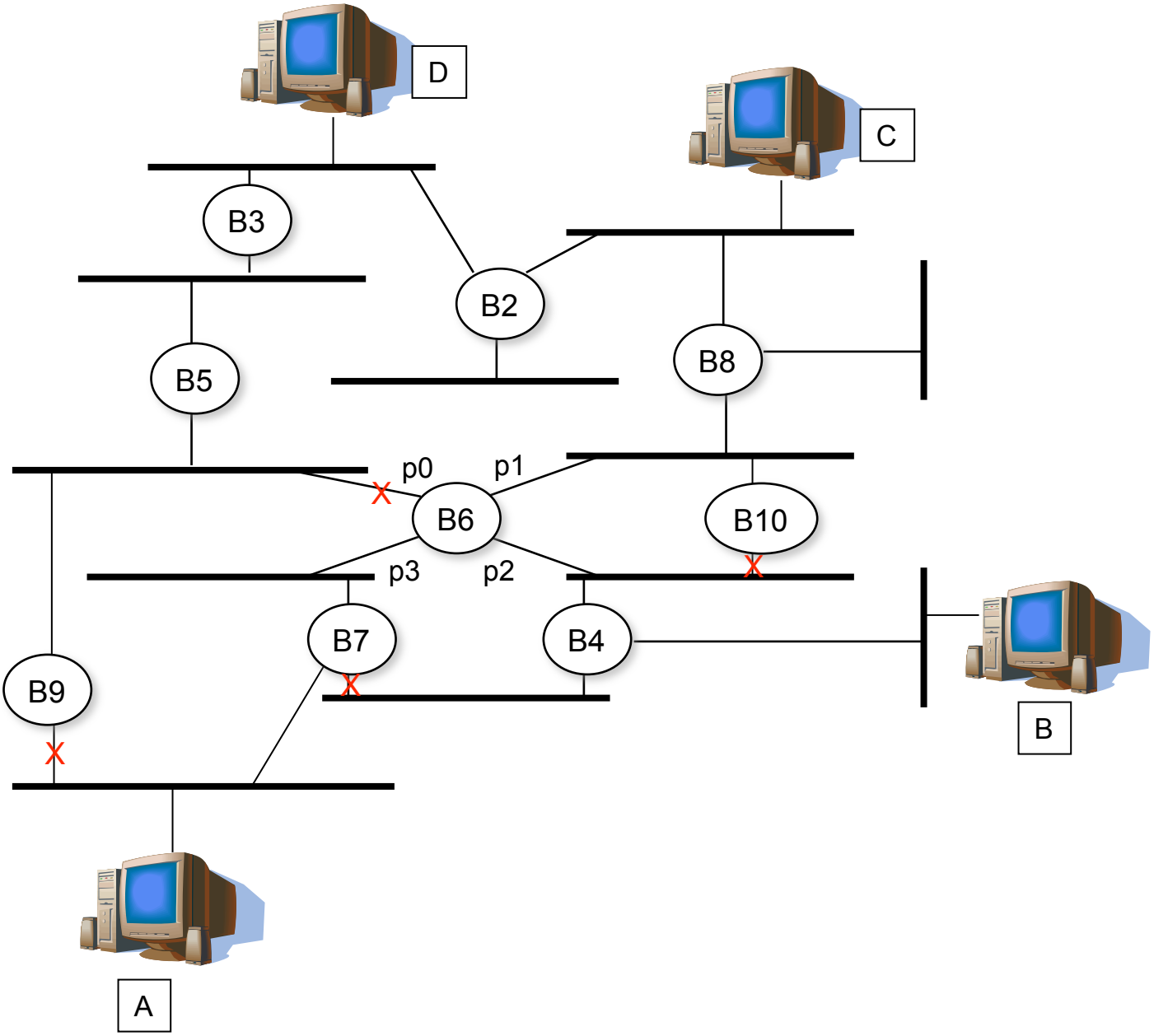
(1) Because when a message are transmitting, all the hosts between the sender and receiver has to wait, otherwise it will cause collision then all the time been wasted. That's why more hosts will cause more collision, lower the throughput;
(2) Because host should detect if there's collision while transmitting, if the hosts are too far, then the propagation delay is too large, the host won't know if there's collision after transmission.

Problem 4 (20 points) Consider a bridged Ethernet shown below. There are 4 computers A, B, C, and D in the network. The ports of bridge B6 are labeled p0, p1, p2, p3 respectively. The bridge IDs are the numeric values. Assume the forwarding tables of all bridges are currently empty.

(a) Indicate which ports are blocked by the bridge spanning tree protocol by putting an "X" over the corresponding ports. (10 points)

(b) With the spanning tree established, suppose A transmits a single packet addressed to B, and C transmits a single packet addressed to D. Explain how the Ethernet bridges forward these packets and how they learn forwarding table entries. Use the space on the next page to write your answer. (7 points)

(c) After the transmissions of the two packets have been completed and the network is idle, what is the content of the forwarding table at bridge B6? (3 points)



(blank page for answer)

My Answer:

(b) For A to B: msg go to B7, then forward to B6, then forward to B4, B10 and B8. And found host B through B4, but msg will still transmit to B2, B3, B5, B9 through B8. B7 will learn host A is on its ‘lower’ port, B6 will learn host A is on its p3 port, B4 will learn host a is on its ‘upper’ port, B8 will learn host A is on its ‘lower’ port, B2 will learn host A is on its ‘upper right’ port, B3, B5 and B9 will learn host A is on its ‘upper’ port;
For C to D: msg go to B2 and B8, then find it host D from B2, but the msg will still go to B10, B6, B7, B4 through B8. B2 will learn host C is on its ‘upper right’ port, B3, B5, B8, B10, B7, B4 and B9 will learn host C is on its ‘upper’ port, B6 will learn host A is on its p1 port

(c)B6 learnt host A is on its p3 port, host C is on its p1 port.

Problem 5 (21 points): Suppose Sammy is designing a new sliding window based reliable transport protocol for use in the Internet. In particular, in Sammy's design, he uses (i) a 4 bit packet sequence number to label each packet, (ii) a retransmission timeout of 20ms, and (iii) a window size of 14 packets.

Provide three detailed reasons in terms of the protocol's correctness or performance to argue that Sammy's design is a poor design.

My Answer:

Reason1: If the window size is 14 packets, you should have 2*14=28 different codes, which mens you should have at least 5 packet sequence number.
Reason2: In common situation, the retransmission timeout is too long, it should been determined by RTT.
Reason3: In common situation, the window size is too large.