CS/ECpE 5516, Spring 1998, Midterm Exam

75 minutes, 120 points
Prof. Abrams

Write your answers on your own paper; a stapler and scrap paper are available at the front of the room. Arrange the answer pages in problem order.

The exam is open Bertsekas and Gallagher. You may also use the handout "**Summary of Post-1987 TCP Specification,"** and you may have written whatever other notes you wish on this handout.

Part 1: Short Problems.

There are 3 problems, each worth 10 points, for a total of 30/120 points.

1. Bit stuffing uses a flag of 01^n0, where n=6. Suppose a colleague proposed using n=1. Would you tell him/her that n=1 is a good or a bad idea? Give a good technical justification for your answer in three sentences or less.

2. Would a stop and wait protocol be a reasonable choice in transmitting digitized voice conversations over a point to point digital channel? Justify your answer in three sentences or less.

3. Could you write a program to run on a computer network using the socket interface and TCP that consists only of two *clients* that communicate with each other (e.g., there is no server)? Justify your answer in three sentences or less.

Hint: The socket calls are close, write, read, connect, bind, accept, socket, listen.

Part 2. Long Problems.

There are 3 problems, each worth 30 points, for a total of 90/120 points.

4. Because you are about to be a graduate of CS/EE 5516 at Virginia Tech, you go for a job interview at NASA (the U.S. space agency). The first interview question is, "Is TCP/IP a good protocol to use for point-to-point communication in an interstellar communication network, between Earth and planned space stations that will be located at nearby stars?"

Carefully defend your answer. Don't give a single reason to justify your answer, but instead list all pertinent reasons. Clearly state at the start of your solution whether your answer is "yes" or "no."

5. Imagine that you have instrumented the code of a TCP implementation to write to disk the value of the congestion window (CWND) for a certain connection every time that it is assigned a new value. Assume that the receiver window advertisement is infinite.

Listed below are three traces. Each trace starts when the connection is first opened. For each trace, state whether the trace *could* have been produced by the post-1987 TCP spec. If you believe that the trace violates one of these algorithms, explain the violation in three sentences or less.

(For simplicity, we assume that every segment sent has an identical size, such as 1K bytes, and that the numbers below represent multiples of that size (e.g., "2" in trace means congestion window size is 2K).)

a) 1,2,4,8,16,8,4,8,9,10,11,12
b) 2,4,8
c) 1,2,4,8,16,32,1,2,4,8,16,32,64

6. Suppose you attach a network monitor to observe the sequence of packets that travel over a link from node A to node B. Your job is to deduce what protocols, if any, could have generated the observed sequence. Shown below are three unrelated observations. Boxes for A denote SN. All diagrams show the *entire* sequence of messages sent by A. The left edge of each diagram is the start of the connection (e.g., t=0). In D0 to D2, there *may* be traffic from B to A, but the monitor does not display this.

Repeat (a) through (d) below for each of the diagrams below. Use the definition of each protocol that is given in section 2.4 of Bertsekas and Gallagher.

(a) Could stop-and-wait (SW) have generated the diagram? Justify your answer.
(b) Could go-back-n (GBN) have generated the diagram? Justify your answer.
(c) Could selective-repeat (SR) have generated the diagram? Justify your answer.
(d) If you answered **yes** to (b) or (c), what is the strongest conclusion that you can infer about the *value* of n? Justify your answer.

Note: When you write your answer, answer parts (a) to (d) for D0 before moving on to D1, and so on. So your answer should be ordered as follows:
D0: (a), (b), (c), (d)
D1: (a), (b), (c), (d)
D2: (a), (b), (c), (d)

Diagram D0:

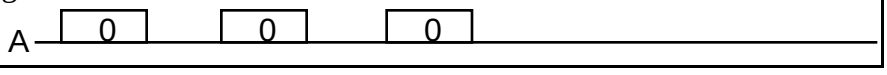


Diagram D1:

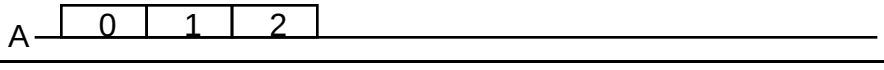
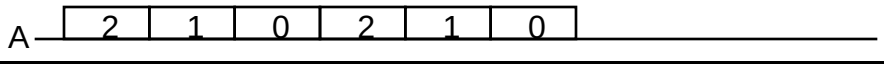


Diagram D2:



Answers:

1. It is a bad idea.

Answer #1: The flag 010 is highly likely to occur in the data field of a frame. If it occurs accounts for p percent of the bits in the data field, then the data field will be elongated by p percent. For large p, this is excessive overhead.

Answer #2: The stuffing rule in Bertsekas and Gallagher says to put a 0 after n-1 bits. If n=1 then the stuffing rule does not make sense.

2. No. Humans listening to a voice conversation must hear a continuous stream of voice, which makes voice traffic different in its timing requirements than data traffic. Use of a stop and wait protocol would introduce gaps into the streams of packets delivered by the receiving DLC to the receiving network layer, because the sending DLC would block until the previous packet is delivered. In this case an occassional lost packet, which could be represented at the receiver as a garbled voice, is prefeable to a gap at the receiver.

3.

Answer #1: Clients like the one we looked at in class are connection-oriented. The client does a *connect* call to block until a connection with the server is initiated. (The server does an *accept* call to wait for a client to request a connection.) If two clients tried to talk to each other, both would block at *connect* forever. Furthermore, a client's (local) is assigned a port number automatically, so there is no way that another client could know the port number to try to connect to it. Finally, a client by definition tries to connect to one specific host (the server), and a server waits for any anonymous host to connect to it (namely a client). This won't hold true if two clients try to talk directly to each other.

Answer #2: Clients that are connection-less (unlike the ones we looked at in class) could communicate if they use the *sendto* and *recvfrom* calls, because they specify the ip address and port number of another process. Note that in the connectionless case the clients do a *bind* call to select a local port number, rather than having a port automatically assigned, avoiding a problem mentioned in answer #1.

4.

Answer: No

The most critical observation is that the round trip delay would take *light years*. As a consequence:

- You could not use an acknowledgment/retransmission strategy for error detection/recovery (Or, as one student put it, the sender could be dead by the time the retransmission timer pops!)
- connection establishment would take light years (use UDP instead of TCP)

- Sending buffer space would be equal to channel bandwidth times the round trip propagation delay - more storage than exists in all computers ever built! This also implies the need for more than a 32 bit sequence number. But IP version 6 provides this.
- Updating routing tables in IP backbones might be based on the network state that existed many light years ago

In addition:
- When interstellar communication is commonplace, the number of hosts will probably be much larger than today's Internet, and thus range of IP addresses required will probably have been exhausted!

Incidentally, a better choice might be UDP with an error correcting (not just detecting) code to avoid connection establishment, release, acknowledgments, and retransmissions. The protocol could use a very large IP datagram size, possibly containing the entire message

Note: The possible corruption of a radio wave through space is not really a problem for TCP/IP, because the protocols are used on radio nets with high loss rates on earth. (Think of the Univ. of Hawaii ALOHA.)

[Grading: +3 for choosing "NO"; +9 for each correct reason. Thus one needs 3 reasons to obtain 3+3*9=30 points. Some students wrote that TCP is designed for point to point communication, not broadcast; this received +3, because I did not say that the interstellar network would carry just broadcast traffic.]

5.
a) Impossible: CWND, when reduced, must take on the value 1. 16->8 violates this.
b) Impossible: CWND is initially 1.
c) Impossible: SSTHRESH is set to CWND/2 whenever CWND is reset to 1. Thus SSTHRESH must be 32/2=16 when CWND goes from 32 to 1. Hence CWND can increase like 1,2,4,8,16, but then must increase by 1/16 afterwards (not to 32 and 64 as the trace shows).

6.

D0:
(a-c) Can be SW, GBN, SR. GBN and SR are possible because A might send 0, then timeout then resend 0, timeout, and resend 0 again, all before A receives a frame from B.
(d) n>=1. Explanation: n must be at least one in any GBN and SR. In D0 we only see one frame being sent. This gives us no upper bound on n. It could be that n=256. Thus n>=1 is the strongest condition.

D1:
(a) Cannot be SW because A cannot send SN=1 until B sends an acknowledgment for SN=0, and this cannot be done in time zero. Can

be GBN or SR. Thus there must be an idle time between sending 0 and 1. A similar argument applies to SN=1 and SN=2.

(d) n>=2, because the ACK for packet 0 (with RN=1) could be received during the transmission of packet 1, thereby advancing SN_{min} to 1 so that packet 2 can be sent as soon as transmission of packet 1 completes.

D2:

(a-c) Cannot be SW for same reason as D1. Can be GBN or SR, because all reverse packets could be delayed so long that A retransmits its three packets. Note that rule 5 does not specify the order in which the *first* copy of each packet is sent; thus the first frame sent by A can be 2 rather than 0. Note that rule 5 on page 76 or 80 do not place any constraint on what order is used for retransmissions.

(d) n>=3

Grading:

(a-c) -1 for each missing element in the list
-3 if "impossible" was given incorrectly, or if "impossible" was omitted

(d) -1 for writing "=" rather than ">="
-1 for D1 for writing "n>=2" not "n>=3"