Computer Networks (CS31006)
Class Test-3, Spring 2020-21
Sample Solution

Students: 155
Date: 20-February-2021
Full marks: 30
Time: 60 minutes
Credit: 20%

INSTRUCTIONS: This is an OPEN-BOOK, OPEN-NOTES test. Please write your answers in a text file/.doc file, convert it to PDF, and submit this PDF file containing ONLY YOUR ANSWERS on Moodle. PLEASE DO NOT SUBMIT SCANNED HAND-WRITTEN ANSWERS, SUCH ANSWER-SCRIPTS WILL NOT BE GRADED. DO NOT FORGET TO WRITE YOUR NAME AND ROLL NUMBER AT THE TOP OF YOUR ANSWER SHEET. ANY DETECTED CASE OF PLAGIARISM WILL BE DEALT WITH STRICTLY, WITH ALL THE IMPLICATED STUDENTS RECEIVING ZERO IN THIS TEST. You may use calculators if required. This question paper contains two pages. ANSWER ALL QUESTIONS.

1. Does TCP use Selective Repeat (SR) or a Go-Back-N (GBN) for flow control? Explain your answer. [2 + 2 = 4]
Answer:
[**Give marks if the students mention that TCP uses SR only, and can explain TCP Flow control by pointing out the ACK mechanism and correlate it with SR]**
TCP essentially uses Selective Repeat (SR) for flow control; however, certain aspects of the TCP acknowledgement mechanism resembles Go-Back-N (GBN).

TCP buffers the out of order segments in the receiver's window. When the receiver gets out of order segments, it sends duplicate acks to the sender. The sender then retransmits the next segment, after which the receiver sends the cumulative ack considering all the already buffered segments. Thus, the dropped segments are selectively retransmitted. In its ability to buffer out-of-order segments, TCP resembles SR; however, in its inability to individually acknowledge out-of-order segments, TCP resembles GBN.

2. Explain the "SYN Flood" attack on TCP connections. Also, suggest one method of resisting the attack. [2 + 2 = 4]
Answer:
In the TCP three-way handshaking for connection establishment, first the client sends a SYN packet. In SYN Flood attack, the attacker keeps sending and flooding the server with many SYN packets, and does not wait for the SYN-ACK response. The TCP server temporarily maintains a new open port connection (backlog) for a certain timeout period for each such SYN packet, thus

often exhausting the limit of such half open connections. Therefore, the server cannot process the legitimate client requests.

TCP typically uses the randomized sequence number, reduced SYN_RECEIVED timer, SYN Cache, SYN Cookies, etc. to resist such attacks.
[**Give marks if the student mentions one of the methods and explains it. One sample explanation is given below**].

One method for resisting the attack is SYN cookies:

1. Server upon receiving SYN packet , drops the SYN packet if the backlog is getting full.
2. Server responds with a cryptographic cookie: SYN queue entry is encoded into the sequence number, and sent in the SYN+ACK response.
3. If the client responds with ACK, from the sequence number, the SYN cookie is decoded by the server and the half open connection is reconstructed.

3. Assume that you have set up a TCP connection over a lossless link with end-to-end bandwidth 2 Gbps. Further assume that you are using 16-bit sequence numbers for individual bytes. If the end-to-end link delay is 50 milliseconds, is it safe to use this 16-bit sequence number field for a sliding window based flow control algorithm?
[Hint: By "safe", we mean the ability of the protocol to distinguish between different segments. Further, note that we can use a simple sliding window protocol and not an ARQ protocol, as the link is lossless.] [3]
Answer:
Bandwidth = 2 Gbps
End to end delay = 50 ms

At a time, $2 * 10^9 * 50 * 10^{-3} / 8$ bytes can be in the channel (bdp) = 12500000

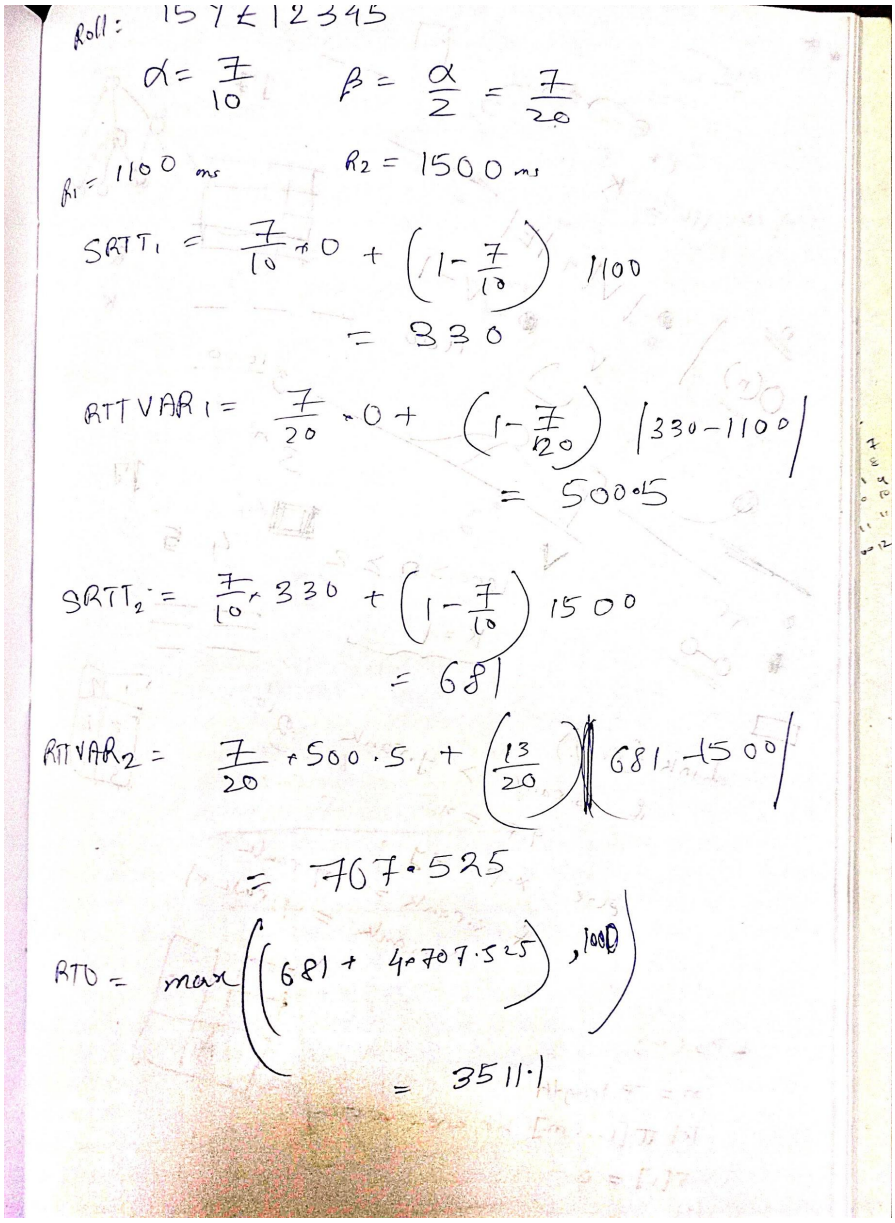
16 bits seq number has the range of 0 to 65535

Therefore it is not safe.

(Even if we account for the header, and 1500 MTU, the seq number required would be 12500000 - (12500000 / 1500 * 20) = 12333340)

4. Explain the Silly Window Syndrome of TCP, and Clark's Solution to solve it. [2 + 2 = 4]
Answer:
Suppose the receiver application reads data very slowly 1 byte at a time. Therefore, the receiver buffer becomes full , and one byte is freed after the application performs a read. Now a window update segment is sent with window size 1. So the sender then sends 1byte of data after which the receiver buffer again becomes full. This is silly window syndrome.
Clark's solution: receiver does not send window update segment when 1 byte is freed. It waits until sufficient receiver buffer is free (e.g. at least half the window size)

5. Consider a TCP connection, which calculates the TCP *Retransmission Timeout* SRTT, RTTVAR and RTO parameters following the standard *Jacobson's Algorithm*, with parameters α and β determined as follows. The parameter α can be calculated from your roll number, by adding all the letters and digits, calculating the remainder modulo-5, dividing by 10, and adding to $\frac{1}{2}$. The letters are assumed to have the following numerical values: A=1, B=2, $\cdots$ Z=26. For example, if your roll number is 15YZ12345, then $\alpha = \frac{(1+5+25+26+1+2+3+4+5) \pmod 5}{10} + \frac{1}{2} = \frac{2}{10} + \frac{1}{2} = \frac{7}{10}$. The parameter β is set to be $\beta = \frac{\alpha}{2}$. Assume the following initial values: $SRTT_{\text{initital}} = 0$ ms and $RTTVAR_{\text{initial}} = 0$ ms. Given that the first two measured values of the round-trip time are 1100 milliseconds and 1500 milliseconds respectively, and your roll number, calculate the value of the *RTO* parameter after the acknowledgement for the second TCP segment has been received. **[5]**



6. State whether each of the following statements is True or False, with a brief (1-2 sentence(s)) explanation in support of your answer:

(a) Ordinary implementations of TCP uses Selective Acknowledgement (SACK) to request for missing segments. [2]

True
SACKs allow a receiver to acknowledge non-consecutive data, so that the sender can retransmit only what is missing at the receiver's end.

(b) The Maximum Segment Size (MSS) for a TCP connection is often set to about 1500 bytes in practice. [2]

True
MSS is limited to 1500 because of the MTU of the link. This results in a max amount of data to be 1460 bytes.

(c) During the Slow Start phase, the CWnd parameter in TCP Tahoe increases at an extremely slow rate. [2]

False
CWnd is doubled at every RTT. This results in an exponential increase in the rate.

(d) Delayed Acknowledgement combined with Nagle's Algorithm can result in starvation in a TCP connection. [2]

True
In Delayed Acknowledgement, the Receiver waits for data before sending ACK, whereas in Nagle's algorithm the sender waits for ACK.

(e) The "URG" flag has lower priority than the "PSH" flag in TCP. [2]

False.
URG flag is even more aggressive since it allows out of order data delivery to the receiver even when the receiver window is 0.
PSH makes the sender prepare and send a segment immediately.