Department of Electrical Engineering and Computer Sciences
University of California at berkeley
EECS 122 : Introduction to Communication Networks
Spring 2005 - Assignment 4 Solution – Available: April 19, 2005

Problem 1 [10 points]

(a) Peterson and Davie, Chapter 5, Exercise 8. [5 points]

Solution

The sequence number does not always begin with a 0. Since it is randomly generated, the initial sequence number might be close to the maximum sequence number, and hence may wraparound even after sending a small number of bytes.

(b) Peterson and Davie, Chapter 5, Exercise 11. [5 points]

Solution

The problem here is that, since TCP is reliable, there is no way of determining whether a packet arrived on the first attemptor whether it was lost and retransmitted.

Problem 2 [10 points]

Peterson and Davie, Chapter 5, Exercise 12.

Solution

(a) [5 points]

The bandwidth in bytes/second is 125MB/s. The sequence numbers wrap around when we send 2^{32} bytes = 4GB. This would take 4GB/(125MB/s) = 32 seconds.

(b) [5 points]

The increment occurs one every 32 ms. Therefore, it would take about $32 \times 4 \times 10^9$ ms ≈ 4 years, for the timestamp field to wrap.

Problem 3 [15 points]

Peterson and Davie, Chapter 5, Exercise 19, part a.

Solution

The following sequence may occur:

1) C connects to A, and gets A's current clock-based ISN_{A1}.
2) C sends a SYN packet to A, purportedly from B. A sends SYN+ACK, with ISN_{A2} to B. This packet is ignored by B.
3) C makes a guess at ISN_{A1}, e.g., ISN_{A1} plus some suitable increment, and sends the appropriate ACK to A (along with, maybe, some malicious data). This packet also has the forged source address of B.
4) C does nothing further, and the connection either remains half-open indefinitely or else is reset, but the damage is done.

Problem 4 [20 points]

Peterson and Davie, Chapter 5, Exercise 20.

Solution

(a) [8 points]

T=0, ‘a’ sent.

T=1s, ‘b’ collected in buffer.

T=2s, ‘c’ collected in buffer.

T=3s, ‘d’ collected in buffer.

T=4s, ‘e’ collected in buffer.

T=4.1s, ACK of ‘a’ arrives, ‘bcde’ sent.

T=5s, ‘f’ collected in buffer.

T=6s, ‘g’ collected in buffer.

T=7s, ‘h’ collected in buffer.

T=8s, ‘i’ collected in buffer.

T=8.2s, ACK of ‘bcde’ arrives, ‘fghi’ sent.

(b) [6 points]

The user would type blindly at times. Characters would be echoed between 4 and 8 seconds late, and echoing would come in chunks of four or so.

(c) [6 points]

With the Nagle algorithm, the mouse would appear to skip from one spot to another (because multiple mouse positions may be clubbed together in the same segment for some time before actually being sent). Without the Nagle algorithm the mouse cursor would move smoothly (because the mouse positions are being sent regularly as soon as they are provided), but it would display some inertia: it would keep moving for one RTT even after the physical mouse was stopped.

Problem 5 [15 points]

Peterson and Davie, Chapter 5, Exercise 25.

Solution

If every other packet is lost, we transmit each packet twice.

(a) [6 points]

Let $E \geq 1$ be the value for EstimatedRTT, and $T = 2 \times E$ be the value for TimeOut. We lose the first packet and backoff TimeOut to $2 \times T$. Then, when the packet arrives, we resume with EstimatedRTT $= E$, TimeOut $= T$. The TimeOut does not change.

(b) [9 points]

Let T be the value for TimeOut. When we transmit the packet the first time, it will be lost and we will wait for time T. At this point, we back off and retransmit using TimeOut $= 2 \times T$. The retransmission succeeds with an RTT of 1 second, but we use the backed-off value of $2 \times T$ for the next TimeOut. Thus, TimeOut doubles with each received packet.

Problem 6 [10 points]

Peterson and Davie, Chapter 5, Exercise 39.

Solution

Incrementing the ACK number for a FIN is necessary, so that the sender of the FIN can determine that the FIN was received and not just the preceding data.

For a SYN, any ACK of subsequent data would increment the ACK number, and any such ACK would implicitly acknowledge the SYN as well (data cannot be ACKed until the connection is established). Thus, the incrementing of the ACK number for the SYN is not really a design necessity.

Problem 7 [20 points]

Consider the following scenario: a client on machine A is communicating to a server on machine B using a TCP connection over a 2 Mbps satellite link. The round-trip delay from A to B is 1 second.

(a) What should be the optimal window size for the TCP connection? Is it possible to advertise this window size in TCP? If not, suggest an extension to TCP to handle this. You may assume that the round-trip delay is constant. [10 points]

(b) Is it always good to advertise the optimal window size? Justify your answer. [10 points]

Solution

(a) To maximize the throughput, the optimal window size = RTT-Bandwidth product = 2 Mbits = 250 Kbytes. The advertised window size field in TCP header is only 16 bits which corresponds to a maximum window size of 64 Kbytes. Recent TCP extensions, however, take this into account by providing an additional scaling factor for the window size. As an example, if the value in the advertised window field says 4096, and the scaling factor is 32, then the effective window size = 4096×32 bytes = 128 Kbytes.

(b) It is not always good to advertise the optimal window size. One reason is the large traffic that might arise as a result, leading to congestion. This might also clog the resources of the receiver, especially if it is a server handling many TCP flows at the same time. In this case, the total memory required to handle multiple flows might be large as a result.