**Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**CS335 Test #1**

1. For each concept, specify the OSI layer that is principally associated with it:
   1. SMTP *-- Application Layer*
   2. IP addresses *– Network Layer*
   3. Routers *– Network Layer*
   4. Port Numbers *– Transport Layer*
   5. Sockets—*Application Layer*
   6. Login passwords – *Session Layer*
   7. Encryption – *Presentation Layer*
   8. File transfer – *Application Layer*
2. A student writes a client and a server using sockets. The student sends 32-bit integer data between the client and server, but forgets to use the htonl and ntohl calls. Surprisingly, the program works fine for all integers.
   1. Describe the conditions under which this could occur.
      1. *This would occur if both machines are big endian, or if both machines are little endian.*
   2. Under what conditions would it not be possible?
      1. *If one machine is big endian and the other machine is little endian.*
   3. Suppose the student tried to fix this problem, but used htonl where ntohl should have been used. Under what conditions could this work?
      1. *This would always work, as htonl and ntohl simply reverse the order of the bytes on little endian machines, and do nothing on big endian machines.*
3. Describe the protocols used to transfer character strings in each of the following protocols:
   1. http
      1. *Character data is sent as ASCII bytes followed by carriage return and line feed.*
   2. The login program you wrote
      1. *An integer value specifying the number of character in the string is sent, followed by the ASCII characters that make up the string. Null-terminators are not transmitted.*
4. Suppose a go-back-n protocol, with n=10, sends segments 0-9 to a peer

that is currently expecting sequence number 0. However, segments 3 and 8 are

corrupted in transmission. In what order will the receiver accept and

acknowledge the 10 segments?

*GBN receivers reject any segment that does not contain the expected sequence number. Thus the order in which the segments will be accepted and acknowledge is 0,1,2,3,4,5,6,7,8,9.*

1. For the previous question, specify how many times each of the 10 segments will be

transmitted?

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

1. Suppose a selective repeat protocol with a send window=[0..9], sends

segments 0-9 to a peer with a receive window that is currently accepting segments

[0-10]. However, segments 8 and 3 are corrupted in transmission. In what order is

the receiver likely to accept and acknowledge the 11 segments?

0, 1, 2, 4, 5, 6, 7, 9, 3, 8

1. In the previous question, how many times will each of the segments be

transmitted?

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 1 |  |

1. What is the principal disadvantage in using a “stop and wait” protocol? How is this overcome?
   1. *Stop and wait protocols can provide very poor utilization of available bandwidth. A solution is to pipeline the process by sending additional segments before the first segment has been acknowledged. The maximum number of unacknowledged segments that can be in transit is determined by the sender’s window size. Stop and wait protocols can be considered to have a windows size of 1.*
2. Would it be possible to use UDP as a lower-level layer in a reliable, connection-oriented protocol? Explain.
   1. *Yes. TCP/IP is an example of a reliable, connection-oriented protocol that is built using an unreliable, connectionless protocol (IP) as a lower-level layer. UDP could be used just as IP is used in TCP/IP, with an application-layer protocol providing all the capabilities that TCP provides.*
3. It would be nice to have a cs.lynchburg.edu domain that would allow the computer science department to create new domain names for our machines without having to request names from ITR. These machines could then have names such as mintaka.cs.lynchburg.edu and altair.cs.lynchburg.edu. The mechanism that would be used to find the IP addresses for these names would be very similar to the mechanism used to resolve [www.lynchburg.edu](http://www.lynchburg.edu). Briefly explain the process. What steps do you think would be necessary to implement the cs.lynchburg.edu domain?
   1. *The Domain Name System (DNS) is implemented using a hierarchical distributed database that maps domain names to IP addresses. The “root” DNS servers are found at the top of this hierarchy. They provide information about the “top-level” domain names such as “.com”, “.edu”, and “.gov”. A request to resolve* [*www.lynchburg.edu*](http://www.lynchburg.edu) *might begin at a root DNS server. The server would respond with a referral to a “top-level .edu” server. A request to that .edu server would then result in a referral to a lynchburg.edu DNS server that is managed by Lynchburg College. That server would be able to provide an IP address for* [*www.lynchburg.edu*](http://www.lynchburg.edu)*. If the request was for mintaka.cs.lynchburg.edu the same process would take place, but the lynchburg.edu DNS server would refer the request to a DNS server for cs.lynchburg.edu. That system would be managed by the CS department, and would provide the IP address for mintaka.cs.lynchburg.edu. Implementing this would require the establishment of DNS servers for cs.lynchburg.edu and a modification in the lynchburg.edu DNS database that would cause requests to cs.lynchburg.edu to be referred to the CS DNS servers.*
4. What are the principal tasks of the transport layer?
   1. *Demultiplexing – routing traffic that arrives at the machine to the appropriate process/socket running on that machine.*
   2. *Multiplexing – gathering data from different sockets, adding header information to support demultiplexing, and transferring data to the network layer.*
5. Which socket API call is used to specify the transport protocol to use?
   1. *socket()*
6. Which socket API call is used to specify the port number a server will be associated with?
   1. *bind()*
7. Why do servers usually employ multiple threads?
   1. *If there was only a single thread, only one client could be handled at a time, and all pending clients would have to wait. A typical approach is to have the main thread wait for the arrival of new clients. When the client arrives, a new thread is created to process the newly arrived client.*