CN Tutorial-11

181105016

1. In our example transport primitives of Fig. 6-2, LISTEN is a blocking call. Is this strictly necessary? If not, explain how a nonblocking primitive could be used. What advantage would this have over the scheme described in the text?
->The listen call could indicate a willingness to establish new connections but not block .When an attempt to connect was made,the caller could be given a signal.It would then execute,say OK or REJECT to accept or reject the connection.In our original scheme,the flexibility is lacking.

2. Primitives of transport service assume asymmetry between the two end points during connection establishment, one end (server) executes LISTEN while the other end (client) executes CONNECT. However, in peer to peer applications such file sharing systems, e.g. BitTorrent, all end points are peers. There is no server or client functionality. How can transport service primitives may be used to build such peer to peer applications?
->Since the two end points are peers,a separate application-level mechanism is needed that informs the end points at the run time about which end will act as server and which end will act as their addresses.One way to do this is to have a separate coordinator process that provides this information to the end points before a connection between the end points is established.

3. In the underlying model of Fig. 6-4, it is assumed that packets may be lost by the network layer and thus must be individually acknowledged. Suppose that the network layer is 100 percent reliable and never loses packets. What changes, if any, are needed to Fig. 6-4?
->The dashed line from passive establishment pending to established is no longer contingent on an acknowledgement arriving. The transition can happen immediately. In essence,the passive establishment pending state disappears,since it is never visible at any level.

4. In both parts of Fig. 6-6, there is a comment that the value of SERVER PORT must be the same in both client and server. Why is this so important? (Please refer program given Tanenbaum Chapter 6 5th Edition)
->If the client sends a packet to serverport and the server is not listening to that port,the packet will not be delivered to the server.

5. In the Internet File Server example (Figure 6-6), can the connect() system call on the client fail for any reason other than listen queue being full on the server? Assume that the network is perfect.
->The connect() may fail if the server hasnt yet executed its listen() call.

6. One criteria for deciding whether to have a server active all the time or have it start on demand using a process server is how frequently the service provided is used. Can you think of any other criteria for making this decision?
->One other criteria is how the clent is affected by extra delay involved in process server technique.The server for the requested service has to be loaded and probably has to be initialized before the client request can be serviced.

7. Suppose that the clock-driven scheme for generating initial sequence numbers is used with a 15-bit wide clock counter. The clock ticks once every 100 msec, and the maximum packet lifetime is 60 sec. How often need resynchronization take place
(a) in the worst case?
(b) when the data consumes 240 sequence numbers/min?
->a)clock takes 32768 ticks i.e 3276.8 sec to cycle around.At zero generation rate ,the sender would enter the forbidden zone at 3276.8 – 60 =3216.8 sec.
b)At 240 sequence numbers/min,the actual sequence number is 4t, where t is in sec. The left edge of

the forbidden region is 10(t-3216.8).Equating these 2 formulas,we find that they intersect at t=5361.3 sec.

8. Why does the maximum packet lifetime, T, have to be large enough to ensure that not only the packet but also its acknowledgements have vanished?
->looking at the second duplicate packet in fig6-11(b),when that packet arrives it would be a disaster if acknowledgement to y were still floating around.

9. Imagine that a two-way handshake rather than a three-way handshake were used to set up connections. In other words, the third message was not required. Are deadlocks now possible? Give an example or show that none exist.
->Yes, they are. This situation occurs when, for example, host 2 receives an old duplicate "Connection Request" from host 1 and consequently acknowledges it but this "ack" message is lost so host1 will never know that host2 is open for any data coming from host1. The same situation also happens in the opposite direction. Finally both host1 and host 2 will be open for incoming data but neither host will know about the waiting status of the other one.

10. Imagine a generalized n-army problem, in which the agreement of any two of the blue armies is sufficient for victory. Does a protocol exist that allows blue to win?
->No, because the two army problem is unsolvable due to arbitrary communication failures and unreliable links. It is the same for two or more armies.