P2 –A networking program usually has two programs, each running on a different host, communicating with each other. The program that initiates the communication is the client. Typically, the client program requests and receives services from the server program.

P3 - In the OSI protocol model, physical communication between peers takes place only in the lowest layer, not in every layer

P4 – Message and byte streams are different.

In a **message stream**, the network keeps track of message boundaries. In a **byte stream**, it does not.

For example, suppose a process writes 1024 bytes to a connection and then a little later writes another 1024 bytes. The receiver then does a read for 2048 bytes.

With a message stream, the receiver will get two messages, of 1024 bytes each.

With a byte stream, the message boundaries do not count and the receiver will get the full 2048 bytes as a single unit. The fact that there were originally two distinct messages is lost.

P5 –Negotiation has to do with getting both sides to agree on some parameters or values to be used during the communication. Maximum packet size is one example, but there are many others.

P6 – Both models are based on layered protocols. Both have a network, transport, and application layer. In both models, the transport service can provide a reliable end-to-end byte stream. On the other hand, they differ in several ways.

The number of layers is different, the TCP/IP does not have session or presentation layers, OSI does not support internetworking, and OSI has both connection-oriented and connectionless service in the network layer.

P7

A) Data link layer

B) Network layer

P8 – Among other reasons for using layered protocols, using them leads to breaking up the design problem into smaller, more manageable pieces, and layering means that protocols can be changed without affecting higher or lower ones.

P9 – TCP is connection oriented, whereas UDP is a connectionless service.

P10 – Connection-oriented communication has three phases.

1) In the establishment phase a request is made to set up a connection.

2) Only after this phase has been successfully completed can the data transfer phase be started and data transported.

3) Then comes the release phase.

Connectionless communication does not have these phases. It just sends the data.

P11 – Frames encapsulate packets. When a packet arrives at the data link layer, the entire thing, header, data, and all, is used as the data field of a frame. The entire packet is put in an envelope (the frame), so to speak (assuming it fits).

P12 – With **n** layers and **h** bytes added per layer, the total number of header bytes per message is **hn**, so the space wasted on headers is **hn**. The total message size is **M+nh**, so the fraction of bandwidth wasted on headers is **hn/(M + hn)**.

P13 – If the network tends to lose packets, it is better to acknowledge each one separately, so the lost packets can be retransmitted.

On the other hand, if the network is highly reliable, sending one acknowledgement at the end of the entire transfer saves bandwidth in the normal case (but requires the entire file to be retransmitted if even a single packet is lost).

P14 – In a NAK only protocol, the loss of packet **x** is only detected by the receiver when packet **x+1** is received.

That is, the receivers receives **x-1** and then **x+1**, only when **x+1** is received does the receiver realize that **x** was missed.

If there is a long delay between the transmission of **x** and the transmission of **x+1**, then it will be a long time until **x** can be recovered, under a NAK only protocol.

On the other hand, if data is being sent often, then recovery under a NAK-only scheme could happen quickly. Moreover, if errors are infrequent, then NAKs are only occasionally sent (when needed), and ACK are never sent – a significant reduction in feedback in the NAK-only case over the ACK-only case.

P15 – First of all, reliable communication (in our sense, that is, acknowledged) may not be available. For example, Ethernet does not provide reliable communication.

Packets can occasionally be damaged in transit. It is up to higher protocol levels to deal with this problem.

Second, the delays inherent in providing a reliable service may be unacceptable, especially in real-time applications such as multimedia. For these reasons, both reliable and unreliable communications coexist.