**CECS 327 Assignment 4 - Communication**

**20 points**

1. Why are transport-level communication services often inappropriate for building distributed applications?

Transport level communication services are inappropriate because the developers of the application end up having to come up with proprietary solutions to problems when dealing with other applications.

1. Assume a client calls an asynchronous RPC to a server, and subsequently waits until the server returns a result using another asynchronous RPC. Is this approach the same as letting the client execute a normal RPC? What if we replace the asynchronous RPCs with asynchronous RPCs?

No, this approach is not the same. An asynchronous RPC returns an acknowledgment to the caller and another message is sent over the network. Two asynchronous RPCs can be the same if the communication is perfect all the time.

1. Would it be useful to also make a distinction between static and dynamic RPCs? Why?

Yes because it would allow for more flexibility, but a lot of the transparency will in the process.

1. Describe how connectionless communication between a client and a server proceeds when using sockets.

Both the client and the server make a socket and the server binds the socket to a local endpoint. The server does a blocking read call and waits for the incoming data. Also, after making the socket the client makes a blocking call to write data to the sever.

1. Suppose that you could make use of only transient asynchronous communication primitives, including only an asynchronous receive primitive. How would you implement primitives for transient synchronous communication?

Using send primitive in an asynchronous communication the caller would continuously watch for an incoming response from the server. Assuming that the OS stores messages in a local buffer you could block the caller until a signal is received form the OS regarding the message’s arrival. The caller then does an asynchronous receive

1. With persistent communication, a receiver generally has its own local buffer where messages can be stored when the receiver is not executing. To create such a buffer, we may need to specify its size. Give an argument why this is preferable, as well as one against specification of the size.

Making the user pick the size makes implementation easier, but also makes a problem if the buffer fills up because messages will be lost. You could instead make a buffer that can grow and shrink as it is needed making more room for messages, but more work on the system’s end.

1. Give an example where multicasting is also useful for discrete data streams.

Anytime you would need to push a massive update to multiple mirror sites for software distributions or some kind of large online service.

1. How could you guarantee a maximum end-to-end delay when a collection of computers is organized in a (logical or physical) ring?

You could make a token that made sure that data is being sent in the same direction as it is. At each stop you could make a computer hold onto the token for a specified amount of time.

1. How could you guarantee a minimum end-to-end delay when a collection of computers is organized in a (logical or physical) ring?

In order to do this you would need to buffer packets as long as needed. Making the buffer on the end of the receiving party will make sure that less problems interfere with the delivery. This can be taxing on buffer space and capacity needs to be taken into account.

1. Despite that multicasting is technically feasible, there is very little support to deploy it in the Internet. The answer to this problem is to be sought in down-to-earth business models: no one really knows how to make money out of multicasting. What scheme can you invent?

With the invention of 4k streaming and soon 8k, the big ISPs that have been milking the dollar of the consumer are going to have to do something soon. Currently the are able to provide the speeds promised sometimes, but as more people begin to pay for the 1Gb+ internets they are going to be forced to multicast if they want to avoid breach of contract lawsuits.