INF 1060 THEORY QUESION ANSERS

1.**Ved bruk av TCP er det viktig Â lage en egen protokoll**

(applikasjonslags-protokoll)

pÂ toppen av Berkeley socket laget. Hvilke egenskaper

har TCP som gj¯r denne protokollen sÂ viktig?

THESE ARE THE IMPORTANT PROPERTIES OF TCP

-TCP offers a connection-oriented service. The connection is established on both the client and the server side. Once the connection is established, they can interact by exchanging data.

-With TCP there is a guarantee that the data will reach its destination precisely in time. The data will reach its it's destination in the right order and there will be no duplicate.

Incase of anomalies it will be corrected

-Another reason that makes TCP an important protocol is that it breaks up data into packets when sending. This is so because data is a stream of bytes know as TCP segments. This is possible because it a low level protocol unlike higher layer protocol which send data in blocks.

-TCP allows multiple connection and connections are recognized by sockets of both devices. This allows each device to have several other devices independently of each other.

-On the other hand TCP has a good thorough on LAN..

-TCP also provide concurrent transfer of data on both direction, that's to and fro client and vice verse.

2. Hordan ser din applikasjonslags-protokoll ut? Begrunn designet!

My application protocol layer is in such a way that I make request to the server. The request has to be recognized by the server. The message that I send is a string and once I

send the string known then the server sends back reply of the request. I generally used read and send on both side to facilitate interaction. I used this approach because with TCP, there is a guarantee that my request will reach its destination and in the right time. I also choose this protocol because TCP facilitates interaction. Both the array send and receive data have the capacity of 9000. My assumption was the data would not exceed that amount.

3.Ta hensyn til maskinarkitekturen! Hva kan gÂ galt? Ville programmet

ditt virke hvis du kompilerer og kj¯rer tjeneren din pÂ en

arkitektur med Big Endian, og klienten pÂ en av Linux-maskinene

i Ifis terminalstuer?

There are two type of byte order, which is depending on CPU architecture and it how the data are stored. The two types are big and little endian.

Big endian the most significant byte of any multi byte data field is stored on the left, which is also the address of the larger field.

Little endian means that the least significant byte of any multi byte data field is stored on the left, which is also the address of the larger field.

Although we have two byte orders, network protocols have settled for one endian type. All of protocol layers in the TCP/IP suite are in big endian.

This is why big endian is also known as network byte order.

My program will have no issue with the server because its big endian but the Linux machine is small endian. For it to work I have to convert byte of the Linux to big endian by using preprocessor macro htohl(). Which byte order from preprocessor order to network order.

Likewise I have to use htohs() for the opposite work. This is what I have done with port number. So my program will compile.

4.Programmet ditt bruker forbindelsesorientiert kommunikasjon, men

nettfilsystemet NFS bruker hovedsaklig forbindelsesl¯s

kommunikasjon. Hvilke problemer unngÂr du i programmet ditt ved Â

bruke forbindelsesorientiert kommunikasjon?

-My program uses a TCP protocol unlike NFS (Network file System) which uses

UDP. One of the biggest challenges NFS faces in performance. It tends to be slow when there is a heavy traffic. But with connection oriented like the my program it accepts every packet.

Another advantage my program will have over NFS is, it will have better congestion control in a scenario where there is network congestion. Due to TCP reliability, Data will be transmitted in segments at a time.

UDP has no congestion control.

-Another advantage my program will have over NFS will be in error detection.

In case the connection breaks or server breakdown, then my program will stop sending data to the server until the connection is re-established. With NFS with breakdown on server, NFS will not realize unlike in my program.

5.Hvilke muligheter tilbyr Linux og Berkeley socket APIen for Â utvikle

tjenere som kan opprettеholde flere forbindelser samtidig, og som kan

kommunisere med flere klienter samtidig? Hvilken av mulighetene har du

valgt og hvorfor?

Linux and Berkley socket API has several methods that are essential for high performing application. These are the polling methods, which consist of poll, epoll and select. These methods allow multiple connections.

I chose select over the others because of portability. Select in platform independent but poll must not be supported by every platform around which network supports

So I have chosen select because of its compatibility in more platforms

Select can also handle the timeouts within the one-nanosecond precision.

But the two other can handle on millisecond precision. It is not directly relevant

In my case but it is an advantage select has over poll and epoll.