www.techiya.in
BCA Solved Assignment 2023-24

Course Code : BCS-052

Course Title : Network Programming and Administration

Assignment Number : BCA(V)/052/Assignment/2023-24

Q1. If a binary signal is sent over a 3-kHz channel whose signal-to-noise ratio is 30 dB, what is the maximum achievable data rate.

Ans: To calculate the maximum achievable data rate over a communication channel, we can use the Shannon-Hartley theorem. The theorem relates the channel's bandwidth, signal-to-noise ratio (SNR), and the maximum achievable data rate. The formula is given by:

C=B*log2(1+SNR)

Where:

- C is the channel capacity (maximum achievable data rate in bits per second).
- B is the bandwidth of the channel in hertz.
- SNR is the signal-to-noise ratio.

Given that the channel bandwidth B is 3 kHz and the signal-to-noise ratio SNR is 30 dB, we need to convert the SNR from decibels to a linear scale before plugging it into the formula.

Step 1: Convert SNR from dB to linear scale

The formula to convert decibels to a linear scale is:

$$\text{SNR}(\text{l} \in \text{ear}) = 10^{\text{SNR}\frac{\text{dB}}{10}}$$

In this case,

$$\begin{aligned}\text{SNR (linear)} &= 10^{\frac{30}{10}} \\ &= 1{,}000\end{aligned}$$

Step 2: Calculate the channel capacity

Now, we can use the Shannon-Hartley theorem formula to calculate the channel capacity:

$$\mathrm{C} = \mathrm{B} \cdot \log 2(1 + \mathrm{SNR(L)})\mathrm{C} = 3{,}000\mathrm{Hz} \cdot \log 2(1 + 1{,}000)$$

Step 3: Calculate the logarithm

$$\begin{aligned} \log2(1 + 1000) \approx &= \log 2(1{,}000) \\ &= 9.97 \end{aligned}$$

Step 4: Calculate the channel capacity

$$\begin{aligned} \mathrm{C} &= 3{,}000 \times 9.97 \\ &= 29{,}910.00 \end{aligned}$$

C=29,910bits per second

The maximum achievable data rate over the 3 kHz channel with a signal-to-noise ratio of 30 dB is approximately 29,910 bps, which is approximately 30 kbps.

Q2. 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.

Ans: Yes, deadlock is possible by setting up a two-way handshake rather than a three-way handshake.

A situation when none of the processes are able to move forward as each is waiting for each other to complete, which introduces a deadlock situation.

A two-way handshake connects two hosts while a three-way handshake connects a host/client and server, both through TCP/IP.

An Example of the two-way handshake deadlock is:

- Host I sends an old duplicate packet Connection Request to Host II.
- Host II receives it and acknowledges it.

- But this acknowledgment message packet doesn't reach Host I as it gets lost.
- Now, Host I can never come to know that Host II accepted the request and is allowing for incoming data from Host I.
- The same situation may also happen from the other end i.e. the acknowledge message packet from Host I gets lost and doesn't reach Host II.
- This creates a deadlock, where both Host I and Host II are open and accepting data coming from each other, but neither host have any idea about the waiting status of one another.

Q3. What is User Security Management? How does it differ from Disk Security Management?

Ans: User Management is an authentication feature that provides administrators with the ability to create users on a system or a service identify and control the state of users logged into the system and even network. User management includes the ability to query and filter users those are currently logged into the system or network, manually log out users, and control user login counts and login times.

The User Management logout capability also provides more secure control over the state of users. For example, when using IP authentication mode, users are identified by the specified IP address until the IP surrogate time expires. If another person were to use that computer before the IP surrogate time expired, they would be treated as the original user. The common solution for preventing this scenario is to decrease the IP surrogate expiry time, causing the user to be challenged more often. Another key benefit of User Management is visibility into active user sessions. Using the Management Console and CLI, administrators can view all active users and filter display data by user, IP address, or realm for easier viewing. This can be useful for identifying the general login status of users or for making real-time decisions such as immediately logging off a user.

Disk Security Management

Disk Management is an activity to manage the drives installed in a computer like hard disk drives (internal and external), optical disk drives, and flash drives. Disk Management activities are like partition drives, format drives, assign drive letters, and

other such related. For disk management can be done either with help of a tool or with a command to manage system disks, both local and remote.

The following are some of Disk Management functions:

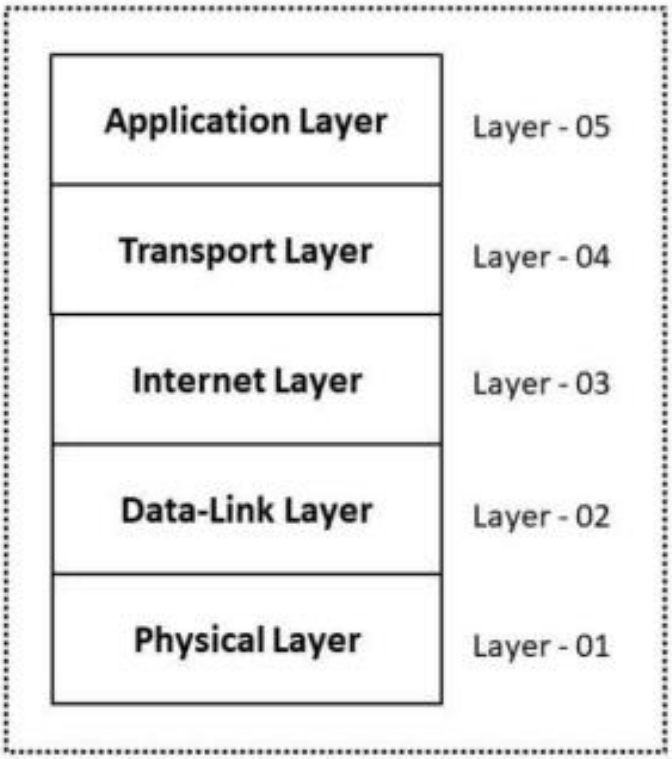
- Create partitions, logical drives, and volumes.
- Delete partitions, logical drives, and volumes.
- Format partitions and volumes.
- Mark partitions as active.
- Assign or modify drive letters for hard disk volumes, removable disk drives, and
- CD-ROM drives.
- Obtain a quick visual overview of the properties of all disks and volumes in the
- system.
- Create mounted drives on systems using the NTFS file system.
- Convert basic disks to dynamic disks.
- Convert dynamic to basic disks, although this is a destructive operation.
- On dynamic disks, create a number of specialty volumes including spanned,
- striped, mirrored, and RAID-5 volumes.

Q4. Explain TCP and UDP architectures.

Ans: TCP Architecture TCP is a connection-based protocol and all communications require that a connection be established first. As this is a relatively expensive operation, it is important to maintain connections across messages within a transaction, or even across transactions. The supervisor process is primarily responsible for managing TCP connections. Each TCP connection has a corresponding application-level connection object and these objects are maintained in a shared hash table. As the figure shows, on receiving a new connection request, the supervisor accepts the connection, creates a new TCP connection object, and adds it to the hash table.

It, however, does not receive or process any SIP messages. Instead, it assigns the new connection to a worker process, which is then responsible for receiving, processing, and forwarding SIP messages that arrive on that connection. The socket file descriptor of the new TCP connection is passed to the worker process using IPC.

Once a worker process receives a new connection from the supervisor process, it receives all messages that arrive on that connection. As TCP is not message-based, that particular worker is the only process that receives messages on that connection. Otherwise, a message might be split across two worker processes. Once a SIP message is received, the worker process processes the message and determines what to do with it. Usually, the worker will determine that it needs to forward the SIP message.

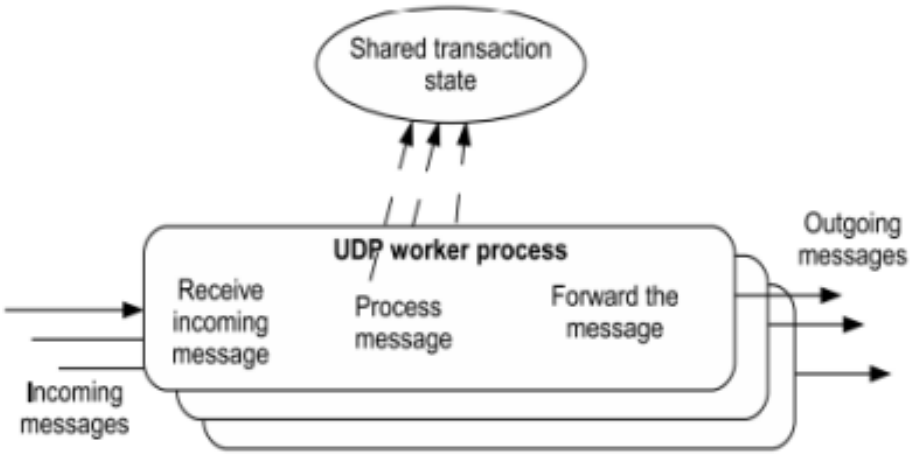


TCP/IP MODEL

Even though a single worker process is responsible for all SIP messages that arrive over a given connection, transaction state still must be shared among worker processes. This is because TCP connections are delegated to a worker process, not SIP transactions. This means that the connections that compose the two halves of the communication in a transaction are likely to be assigned to different worker processes by the supervisor process. The supervisor can- not know ahead of time which connections will form the two halves of a transaction, and this can change over time. Therefore, there are almost always two worker processes involved in every transaction, one for each direction. This requires synchronized access to the shared transaction state.

UDP Architecture

As with the TCP architecture, there are multiple worker processes for handling SIP messages. But unlike the TCP architecture, all the processes are symmetric in nature and there does not exist a supervisor process. Each worker process operates independently to receive, process, and forward SIP messages.



Unlike TCP, UDP is a message-based, connectionless protocol. Since UDP is a connectionless protocol, worker processes do not need to establish connections with the clients. Since UDP is a message-based protocol, worker processes do not need to synchronize in order to receive messages. UDP guarantees that when a worker process attempts to receive a message from the network, it will either receive a SIP message in its entirety or nothing at all. Therefore, each worker process can simultaneously receive messages from the network. In order to process a SIP message, a worker process must first associate the message with a new or ongoing SIP transaction. There is no guarantee, however, that the same worker process will receive all of the messages that make up a single transaction. Therefore, transaction state must be stored in shared memory accessible to all worker processes as shown in figure. Access to this shared memory must be synchronized among the worker processes.

After processing the SIP message, the worker process will typically decide to forward it. With UDP, the worker process can send the complete message out over the network without having to obtain a connection to the client. Similarly, because UDP is message-based, there is no need to synchronize with the other worker processes to send a

message. Specifically, even without synchronization, if two worker processes try to simultaneously send a message, to the same or different clients, neither message will interfere with the other.

As with the TCP architecture, the UDP architecture also includes an additional timer processes. But unlike TCP, the timer process is essential to UDP, since UDP does not guarantee delivery of messages and a stateful proxy must retransmit messages for transactions that do not receive a response. When a message is sent, the worker process creates a new timer which is added to a global list managed by the timer process. The timer process periodically checks the list for timers that have expired. At that point, it accesses the transaction state and retransmits unacknowledged SIP messages. Access to both the global timer list and the shared transaction state must be synchronized with the worker processes.