April 2021 question paper solutions:

**Question 1**

a. Prototype and Meaning of Parameters:

1. Accept System Call:
```
int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
```
Parameters:
- `sockfd`: the file descriptor of the listening socket.
- `addr`: a pointer to a struct sockaddr structure that will hold the address of the accepted connection.
- `addrlen`: a pointer to an integer that contains the size of the sockaddr structure.

Return Value:
- On success, the function returns a new file descriptor that represents the accepted connection.
- On failure, it returns -1 and sets errno to indicate the error.

2. Connect System Call:
```
int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
```
Parameters:
- `sockfd`: the file descriptor of the socket.
- `addr`: a pointer to a struct sockaddr structure that contains the address to connect to.
- `addrlen`: the size of the sockaddr structure.

Return Value:
- On success, the function returns 0.
- On failure, it returns -1 and sets errno to indicate the error.

3. Select System Call:
```
int select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);
```
Parameters:
- `nfds`: the highest-numbered file descriptor in any of the three sets, plus 1.
- `readfds`: a pointer to a set of file descriptors to be checked for readability.
- `writefds`: a pointer to a set of file descriptors to be checked for writability.
- `exceptfds`: a pointer to a set of file descriptors to be checked for exceptional conditions.

- `timeout`: a pointer to a struct timeval that specifies the maximum amount of time to wait for an event to occur.

Return Value:
- On success, the function returns the number of file descriptors that are ready for the requested I/O operations.
- On failure, it returns -1 and sets errno to indicate the error.

b. Relationship between Accept and Connect System Calls:
The `accept()` system call is used by a server to accept a new incoming client connection on a listening socket. When a client attempts to connect to the server, the `accept()` call will block until a connection is established. After the connection is established, `accept()` returns a new file descriptor representing the connection, and the original listening socket is still available for new connections.

On the other hand, the `connect()` system call is used by a client to establish a connection with a server. It initiates a connection request to the server at the specified address and port. If the connection is successful, `connect()` returns 0, and the socket can be used for communication.

Therefore, the `accept()` and `connect()` system calls work together to establish a connection-oriented TCP socket connection on a UNIX system. The server uses `accept()` to wait for incoming client connections and create new sockets for communication, while the client uses `connect()` to initiate a connection request to the server.

c. Use of Select System Call in Non-Blocking Chat Application:
In a non-blocking chat application, the `select()` system call is used to check the status of multiple sockets for readability or writability without blocking. The application can wait for events on multiple sockets and handle them as they occur, without being stuck waiting for one socket to become ready.

For example, if the application is waiting for incoming messages on multiple client sockets, it can use `select()` to check which sockets have data ready to be read. When `select()` returns, the application can loop through the set of file descriptors that are ready for reading and handle the incoming messages appropriately.

Similarly, if the application needs to send messages

**Question 2(A)**

Distance Vector Routing Algorithm (DVR) is a type of routing algorithm used in computer networks, where each router maintains a table of distances to all other routers in the network. However, this algorithm has some limitations that can cause problems in certain situations.

The main problems with DVR include:

1. Slow convergence: DVR takes time to converge to the optimal path after any changes in the network. This delay can cause problems in large networks or when there is frequent network topology changes.

2. Count-to-infinity problem: In a network with a loop, DVR may result in incorrect distance calculations and routing loops. This problem occurs because DVR assumes that routes are either active or inactive, and there is no middle state.

3. Inefficient use of bandwidth: DVR may route traffic over longer paths than necessary, resulting in inefficient use of network bandwidth.

On the other hand, the Link State Routing Algorithm (LSR) is a more advanced routing algorithm that addresses these problems.

LSR works by each router broadcasting information about its links to all other routers in the network, creating a complete network topology map. This map is then used to calculate the shortest path between any two routers in the network.

Some of the benefits of LSR include:

1. Faster convergence: LSR is faster than DVR in converging to the optimal path, as it only needs to recalculate the paths affected by changes in the network topology.

2. Loop-free routing: LSR avoids the count-to-infinity problem by using link-state information, which allows routers to know the exact topology of the network.

3. Efficient use of bandwidth: LSR routes traffic over the shortest path, resulting in efficient use of network bandwidth.

In summary, while DVR is a simple and easy-to-implement algorithm, it has some limitations that can cause problems in certain situations. LSR, on the other hand, provides a more advanced and efficient solution to routing problems in computer networks.

**Question 3:**

a. To subnet the given block of IP addresses into at least 5 subnets, we need to borrow bits from the host portion of the address. We need to determine how many bits to borrow in order to create at least 5 subnets. We can use the formula 2^n >= s, where n is the number of bits borrowed and s is the number of subnets we need. In this case, we need at least 5 subnets, so 2^n >= 5. Solving for n, we get n = 3. This means we need to borrow 3 bits from the host portion of the address to create at least 5 subnets.

The subnet mask that should be used is therefore 255.255.255.224 (/27 in CIDR notation), which is obtained by borrowing 3 bits from the host portion of the default subnet mask (255.255.255.0).

b. To determine the start and end IP address of each subnet, we can use the following steps:

1. Write down the default subnet mask: 255.255.255.0 (or /24 in CIDR notation).
2. Determine the new subnet mask: 255.255.255.224 (or /27 in CIDR notation).
3. Determine the number of subnets created: 2^(number of borrowed bits) = 2^3 = 8 subnets.
4. Determine the number of hosts per subnet: 2^(number of host bits) - 2 = 2^5 - 2 = 30 hosts.
5. Determine the subnet ranges by starting at the network address and incrementing by the subnet size (which is the number of hosts per subnet plus 2).

Using these steps, we can determine the start and end IP address of each subnet as follows:

Subnet 1:
Start IP address: 192.168.0.0
End IP address: 192.168.0.31

Subnet 2:
Start IP address: 192.168.0.32
End IP address: 192.168.0.63

Subnet 3:
Start IP address: 192.168.0.64
End IP address: 192.168.0.95

Subnet 4:
Start IP address: 192.168.0.96
End IP address: 192.168.0.127

Subnet 5:
Start IP address: 192.168.0.128
End IP address: 192.168.0.159

Note that there are three additional subnets that can be created with the remaining IP addresses, but they were not required by the problem statement.

**Question 4**

a. Port numbers are used by transport layer protocols like TCP and UDP to identify the specific application or process that is communicating on a particular device. They are necessary because a single device may have multiple applications that need to communicate with other devices over the network, and port numbers allow for the identification of the specific application or service that should receive incoming network traffic. Ports can only be in the range 0 – 65535 because the TCP/IP protocol suite reserves port numbers in that range for use by applications and services.

b. Connection-oriented transport protocols establish a virtual circuit between communicating devices before data transmission begins, ensuring reliable delivery of data but with increased overhead and latency. Connectionless transport protocols, on the other hand, do not establish a virtual circuit and rely on the receiving device to acknowledge data receipt, leading to less reliable delivery but lower overhead and latency. The advantages and disadvantages of each approach depend on the specific needs of the application.

c. The three primary functions of the network layer in the TCP/IP architecture are:

1. Addressing: The network layer assigns unique IP addresses to each device on the network, allowing for identification and routing of data packets.

2. Routing: The network layer determines the most efficient path for data packets to travel from the source device to the destination device, taking into account network topology, congestion, and other factors.

3. Fragmentation and reassembly: The network layer breaks down large data packets into smaller pieces for transmission over the network, and then reassembles the packets at the destination device.

d. IP addresses are divided into classes based on the first few bits of the address. The five classes are:

Class A: 1.0.0.0 to 126.255.255.255
Class B: 128.0.0.0 to 191.255.255.255
Class C: 192.0.0.0 to 223.255.255.255
Class D: 224.0.0.0 to 239.255.255.255 (reserved for multicast)
Class E: 240.0.0.0 to 255.255.255.255 (reserved for future use)

To determine the class of an IP address, you can look at the first octet (the first set of numbers before the first period). The range of the first octet will determine the class of the IP address.

To extract the network ID from an IP address, you can use subnet masking. The subnet mask is a 32-bit number that is used to divide an IP address into a network address and a host address. By performing a logical AND operation between the IP address and the subnet mask, you can extract the network ID. For example, for the IP address 193.25.14.57 with a subnet mask of 255.255.255.0, the network ID would be 193.25.14.0.

**Question 5**

a. TCP (Transmission Control Protocol) is a reliable, connection-oriented protocol that operates at the transport layer of the OSI model. The TCP header format consists of 20 bytes, which contains several fields that serve different purposes. These fields are as follows:

1. Source port and destination port: These fields are used to identify the sending and receiving processes on the hosts.

2. Sequence number: This field is used to keep track of the bytes of data that are transmitted in a connection. The initial sequence number is randomly generated, and then it is incremented as data is transmitted.

3. Acknowledgment number: This field contains the sequence number of the next byte that the receiver expects to receive. It is used to acknowledge the receipt of the data.

4. Data offset: This field specifies the size of the TCP header in 32-bit words.

5. Reserved: These bits are reserved for future use.

6. Flags: The flags field is used to indicate the status of the TCP segment. The commonly used flags are SYN, ACK, FIN, RST, URG, and PSH.

7. Window size: This field specifies the number of bytes of data that the sender is willing to receive before waiting for an acknowledgment.

8. Checksum: This field is used to verify the integrity of the TCP segment.

9. Urgent pointer: This field is used to indicate the location of urgent data in the segment.

The significance of different fields in the TCP header are as follows:

1. Source and destination port: These fields identify the sending and receiving processes, respectively.

2. Sequence and acknowledgment numbers: These fields ensure reliable data transmission by keeping track of the data sent and received.

3. Flags: These indicate the status of the TCP segment, such as whether it is a SYN, ACK, FIN, RST, URG, or PSH segment.

4. Window size: This field controls the flow of data between the sender and receiver.

5. Checksum: This field ensures the integrity of the TCP segment by detecting errors.

6. Urgent pointer: This field indicates the location of urgent data in the segment.

b. TCP implements congestion control to avoid network congestion by controlling the rate of data transmission. TCP uses several algorithms to provide congestion control, which are as follows:

1. Slow start: This algorithm gradually increases the congestion window until the network becomes congested.

2. Congestion avoidance: Once the network becomes congested, this algorithm reduces the congestion window size to prevent further congestion.

3. Fast retransmit: This algorithm retransmits the lost segment without waiting for a timeout.

4. Fast recovery: This algorithm retransmits lost segments using a reduced congestion window size.

5. Explicit congestion notification (ECN): This algorithm notifies the sender about the congestion in the network, so it can adjust the transmission rate.

c. The AdvertisedWindow and SequenceNum fields of a TCP-like protocol are used to control the flow of data between the sender and receiver. The AdvertisedWindow field specifies the number of bytes of data that the receiver is willing to receive before waiting for an acknowledgment, while the SequenceNum field keeps track of the bytes of data that are transmitted in a connection.

To determine the number of bits that should be included in the AdvertisedWindow and SequenceNum fields, we need to calculate the maximum window size and sequence number, respectively. The maximum window size is determined by the maximum segment lifetime (60 seconds) and the maximum data rate of the network (100 Mbps), which gives us a maximum window size of 7.5 MB. To represent this value in bits, we need to use a 24-bit field (2^24 = 16,777,216 bits).

The maximum sequence number is determined by the maximum window