Tutorial-1

1. What are the importance of computer networking? Compare different client server models and differentiate it with P2P model.

2. Which function is responsible for sending SYN segment during TCP connection establishment phase? Illustrate the TCP states used during connection establishment and connection termination with suitable state transition diagram.

3. Introduce UNIX OS and UNIX network administration along with its file systems and basic administration commands used in UNIX.

4. What is socket and socket API? Explain different socket address structures available in UNIX network programming.

5. What are the ways to pass the length of socket structure for different socket API's argument? Explain them in detail with function prototype and argument detail.

6. The connect () function returns only when the connection is established or an error occurs, what are those errors and when these errors occurred? Explain in detail.

7. Explain different system call in specific order, required to create a TCP client and TCP server in the Unix System.

8. Can we use connect () system call while developing UDP socket? If yes, what will be the outcomes of using connect () in UDP socket? If no, how communicating processes know each other? Explain them in detail.

9. What will happened if you call bind() in tcp client program? Explain the situation with relevant example code where you can use send() and recv() in UDP socket and sendto() and recvfrom() in tcp socket.

10. What is UNIX domain socket and why it is required? Explain the process of passing file descriptor in UNIX.
11. Write short notes on
a) Daemon Process
b) Address conversion functions
c) Signal Handling

Tutorial-2 (Chapter 1, 2 and 3)

1. Why UDP is faster than TCP? Which function is responsible for sending SYN segment during TCP connection establishment phase? Illustrate the TCP different states used in TCP connection and termination with suitable state transition diagram.
2. While establishing TCP connection, the initial sequence number should not start from 0. Why? What is 2MSL wait state?
3. What will happened if you call bind() in tcp client program? Explain the situation with relevant example
code where you can use send() and recv() in UDP socket and sendto() and recvfrom() in tcp socket.
4. Does getaddrinfo() is blocking or non-blocking? List out the different system call used to create TCP and UDP client server on the basis of blocking and non-blocking in nature. Also explain then in very short.
5. What is the purpose of bind () function? What will be the outcomes if we do not specify IP address, port, both, or neither during bind () system call?
6. What is completed connection queue and incomplete connection queue ()? Explain different system call in specific order, required to create a UDP client and UDP server in the Unix System.
7. The connect () function returns only when the connection is established or an error occurs, what are those errors and when these errors occurred? Explain in detail.
8. What is listening socket and connected socket? Explain with function prototype and sample code: when and why getsockname() and getpeername() required?
9. Explain the functions send (), sendto (), recv () and recvfrom (). What are the major differences between wait () and waitpid()? Explain the possible option values that we can supply in waitpid() system call.
10. Differentiate unix domain socket and internet domain socket. Explain the mechanism of passing file descriptor in the Unix System.
11. What are the characteristic of daemon process in unix and how does it started? How do you crate the
daemon process in Unix? Explain.
12. What is signal and why it is required? Explain POSIX signal handling mechanism. Also Explain few signal types.
13. Why IO and IO multiplexing is needed in socket programming? Explain various IO models in Unix system.
14. What is I/O multiplexing? Explain the use of select function in the context of I/O multiplexing in detail.

15. Explain the statement: fork () is called once and it returns twice. What is the significance of calling exec() after fork() system call? What will happen when close () is called in TCP socket during the implementation of concurrent server?
16. What does it mean to be receiving 0 bytes in TCP and UDP socket? How do you get the ip address and port number of connect client and server in TCP and sender and receiver in UDP? Explain with suitable example sample code.
17. What is the basic difference between select() and poll()? Explain poll() and pseelct() function.
18. Differentiate blocking IO model with non-blocking IO model. Can we use signal driven IO as asynchronous IO? If yes, how? If no, why we cannot use signal IO as asynchronous IO?
19. Explain the different types of server? What are the approaches of handling multiple client in Unix system. Explain any one in detail.
20. Give an outline of simple TCP server that can handle multiple clients using Berkeley Socket without using concurrent server. Consider the scenario during data exchange in TCP socket, one site call close() function, what will happened to data in network? Is it still possible to send and receive data packet after calling close() function? What are the ways perform it?
21. How broadcast is different from multicast? Show the sample of code to implement broadcast and multicast.
22. Differentiate fcntl() and ioctl() system call. Explain the meaning of following socket options: SO_BROADCAST,SO_KEEPALIVE,SO_RCVTIMEO, SO_SNDTIMEO, SO_REUSEADDR and SO_LINGER.
23. Why do we need to set and get socket options? Which functions are used to set and get a value of socket options? Explain them in detail with sample code.
24. What are the meanings of SIOCATMARK, SIOCGPGRP and SIOCSPGRP? How do you set socket as non-
blocking using fcntl() and ioctl(). Illustrate with section of code.
25. What is syslogd()? What is the technique for logging messages from a daemon process? Explain with sample code.
26. What is P2P Programming? Explain the challenges of P2P programming.
27. Explain the secure programming workflow. How Hostname Verification is performed? Explain with sequence diagram.

Tutorial-3 (Chapter 4, 5, and 6)

1. How close() is different from shutdown()? Which functions set h_errono global variable? Differentiates errno and h_errno as well.
2. How windows socket is different from berkeley socket? Explain the role of setup(), cleanup() and WSAStartup() function in winsock architecture.
3. Explain winsock architecture. What are the types of dynamic linking in winsock programming?
4. Differentiate blocked I/O with unblocked I/O. Explain all the functions in detail to handle blocked I/O in winsock architecture.
5. What is overlapped I/O? Which function is used to set windows socket in non-blocking mode? Explain non-blocking socket with connect() in the case of winsock architecture.
6. Why WSAGetLastError() is required in winsock programming? Explain WSAGetOverlappedResult() function.
7. In which approach of communication, windows socket provides better functionalities than Unix socket? What will happen if you call shutdown() and close() function? Do these functions are different from WSACleanup() function? Example how/how not.
8. What is the importance of backlog argument in listen() function? TCP sockets are full-duplex sockets. Is it possible to convert those sockets into half-duplex sockets? If possible-show the mechanism/if impossible-explain why.
9. Is to possible to write a common network application that runs in both LINUX and Windows? How would you do it. Show simple example program as well.
10. In which situations asynchronous IO is preferred than synchronous IO? Explain different Asynchronous IO functions in winsock programming.
11. What is event driven programming? Explain winsock extension function WSAPoll() and WSAEventSelect()
12. How do you implement stream communication in Winsock? Describe each step with the help of relevant APIs.
13. Explain with the help of pseudo-code the use of accept with select such that the accept function doesn't block.
14. Why iperf is used? Write short notes on: telnet, tftp, ping, ifconfig/ipconfig, netstat, rlogin.
15. Differenttiate HTTP with websockets. Write a simple server program to demonstrate the concepts of websocket.
16. What is the significance of IDL in gRPC? Explain the working mechanism of gRPC.

17. Explain the key concepts of Software-Defined Networking (SDN). How does SDN architecture separate the control plane from the data plane, and what are the major advantages of this separation in modern networks?
18. What is the OpenFlow protocol in the context of SDN? Discuss how OpenFlow enables communication between the SDN controller and network devices. Also, briefly differentiate between OpenFlow, P4, and Frenetic in terms of programmability and control over network behavior.

Model Questions

SET A

1. a) Why is TCP considered connection-oriented? Justify using its state transition diagram and discuss how each state impacts data flow contro

l. b) UDP lacks reliability guarantees, yet it is widely used in multimedia applications. Explain why, and design a basic protocol logic (pseudo-code or snippet) that adds reliability over UDP.

2. a) Why do we need different types of socket address structures like sockaddr_in, sockaddr_un, and sockaddr_in6? Illustrate their use cases with brief C code snippets.

b) How does incorrect byte ordering affect socket communication between machines of different architecture? Demonstrate how functions like htons() and ntohl() resolve this issue in a TCP program.

3. a) A Unix server needs to handle SIGINT without terminating. Explain how signal handling modifies control flow, and demonstrate using sigaction() in a simple socket program.

b) Why are daemon processes preferred in network services? Outline the steps to daemonize a server process and show relevant code to detach from terminal and run in background.

4. a) Suppose you are building a chat server with multiple clients. Why is select() preferred over fork() in high-load environments? Write a code sketch to illustrate select()-based socket multiplexing.

b) Blocking vs. Non-blocking I/O - explain their conceptual differences with socket system call examples. When would non-blocking I/O be detrimental in network applications?

5. a) Your server needs to reuse a port quickly after crashing. How does setsockopt() with SO_REUSEADDR help? Write a short TCP socket server snippet with this option.

b) Discuss how broadcasting and multicasting differ in socket-level implementation. Demonstrate how to send a multicast UDP packet from a sender program in C.
6. a) Winsock programming requires initialization and cleanup. Explain the lifecycle of a Winsock application and implement a basic TCP echo server in Winsock with proper structure.

b) How does error handling differ between Unix and Winsock socket APIs? Show code fragments of error handling using WSAGetLastError() and contrast with perror() or errno.

7. a) A developer uses select() with Winsock but experiences poor performance. What alternatives exist in Winsock for asynchronous I/O? Explain WSAEventSelect() with an example.

b) Why is multithreading important in network applications? Show how to implement a threadbased TCP server in Winsock that handles each client in a separate thread.

8. a) TLS/SSL provides security over sockets. Explain the role of certificates and handshake in securing communication. Outline the steps to integrate OpenSSL into a Unix TCP server.

b) Compare WebSocket and gRPC with respect to real-time bidirectional communication. Design a simple use case where gRPC is preferable over traditional REST or socket programming.

9. Software-Defined Networking (SDN) centralizes control but introduces new challenges. Explain how OpenFlow changes traditional network flow behavior and why it matters in programmable networks.

10. You're asked to troubleshoot network performance on a server. Choose any five of these tools and briefly explain their use: ping, netstat, iperf, telnet, ifconfig, traceroute, nmap.

Set B

Q1. a) Peer-to-peer (P2P) systems are often more scalable than client-server models. Why? Analyze the tradeoffs between them with real-world examples.

b) How does the layering in TCP/IP differ from the OSI model, and why does this matter for socket programming? Provide concrete programming-level implications.

Q2. a) Why do we use getaddrinfo() over gethostbyname() in modern socket programming? Show its use in a TCP client program that connects to a hostname.

b) Unix domain sockets are local and faster than network sockets. Explain their use in IPC and show how to pass file descriptors between processes.

Q3. a) Suppose you are developing a server that handles SIGCHLD to prevent zombie processes. Explain how signal handlers and waitpid() solve this. Demonstrate with a server skeleton using fork().

b) Explain how sockaddr_storage helps create protocol-independent network applications. Give example code where both IPv4 and IPv6 clients connect to the same server.

Q4. a) Your application needs low-latency communication. Justify when asynchronous I/O is superior to multiplexing. Illustrate using aio_read() or signal-driven I/O in a C sketch.

b) Design a thread-pool-based TCP server using pthread to handle thousands of connections efficiently. Discuss advantages over fork() in terms of performance and resource usage.

Q5. a) You need to log errors and client IPs in your Unix server. Show how syslog() works and write a snippet that logs different severity levels (INFO, ERROR).

b) A client should automatically reconnect if the server is down temporarily. What socket options and programming logic will support this behavior? Justify and show partial implementation.

Q6. a) Implement a UDP Winsock client that sends a message to the server and prints the server's reply. Also discuss the key differences with Unix UDP sockets.

b) How does WSAStartup() manage versioning and resource allocation in Winsock? Provide example output and its implications in a client-server app.

Q7. a) Nonblocking sockets in Winsock improve responsiveness. Demonstrate how to check for WSAEWOULDBLOCK during recv() and handle it correctly in code.

b) Describe how to use WSAEventSelect() to implement event-driven TCP servers. Show its integration in a loop that handles multiple clients.

Q8. a) Compare the data serialization formats used by gRPC (e.g., Protocol Buffers) vs REST (JSON). Why does this difference matter in real-time systems?

b) Secure communication is required in a client banking app. Discuss how SSL certificates, encryption, and socket integration ensure data protection. Mention limitations.

Q9. Your organization is shifting to SDN-based architecture. Explain how programmable switches using P4 language can optimize traffic flows. Give a use case.

Q10. A system admin finds packet loss in a network. Using iperf, netstat, and traceroute, how would you isolate the root cause? Describe practical usage for each tool.