Shameen Dilusha
GDSE67

1. What Is Socket?

- Socket is an endpoint of communication between two programs running on a network. It allows programs to send and receive data to each other.

- Socket is a programming interface

- Sockets work using a client-server model.

- Sockets form the foundation for many common network protocols and services like web servers, email etc.

2. What Is Multiprogramming?

- Multiprogramming is a technique in operating systems that allows multiple processes to be loaded into memory and executed concurrently by the CPU.

- The main advantage of multiprogramming is that it keeps the CPU busy at all times, as opposed to having it idle when processes are waiting for events.

- Overall, multiprogramming improves efficiency on multi-tasking operating systems.

3. Explain the Advantages of Java Sockets & Disadvantages of Java Sockets.

Advantages of Java Sockets:

- Platform independent - sockets work across different operating systems
- Reusability - socket classes and APIs can be reused easily
- Built-in - socket classes are built into Java's core libraries

- Portable - socket code can be reused across different systems
- Flexible - support both TCP and UDP
- Scalable - can handle large volumes of data and high loads

Disadvantages:

- Complexity - sockets have a steep learning curve, code can get complex
- Resource intensive - socket programming requires more resources
- Blocking mode - by default read/write calls are blocking
- Security issues - requires careful handling to avoid vulnerabilities

4. Explain the difference between TCP and UDP protocol?

TCP	**UDP**
● Connection-oriented	● Connectionless
● reliable, provides error checking	● does not
● has in-order guaranteed delivery	● does not guarantee packet order
● has flow control	● does not
● establishes a connection first before data transfer	● does not
● heavier due to error checking hence slower	● lightweight & faster
● used for apps that require high reliability but can tolerate delays.	● for time-sensitive apps that need fast speed.

5. Brief on Client vs Server.

Client	**Server**
● initiates requests	● listens to requests and responds
● requests concurrently	● typically handle multiple
● contact	● using IP address and port number
● use ephemeral ports	● have fixed well-known ports
● Communication is initiated by clients	● accept connections
● only interact with the server	● interact with multiple clients
● web browser is client Email client is client	● web server is server mail server is server

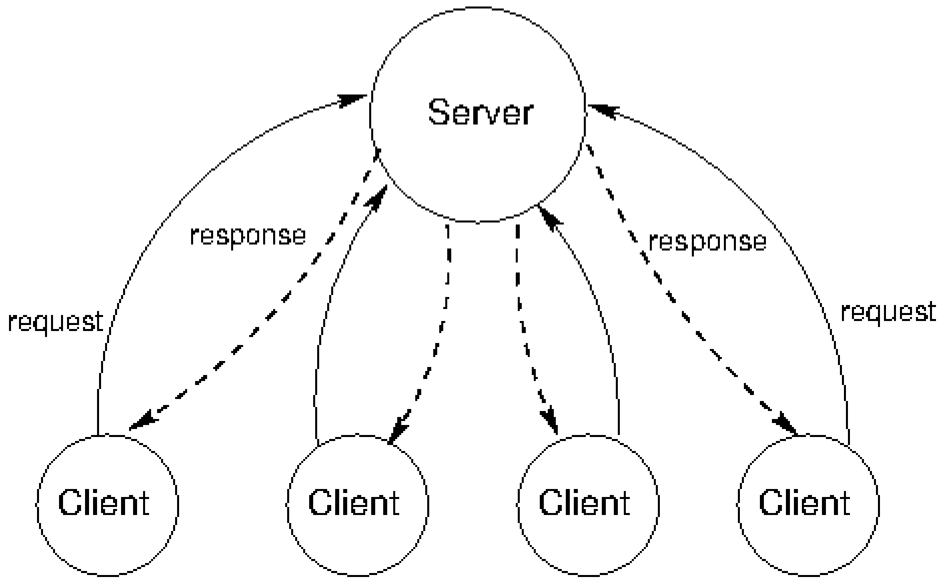
6. Explain the differences between Client and Server.

Client	**Server**
● initiates communication	● waits and responds to requests
● typically interact with one server	● handle requests from multiple clients
● only care about their individual requests	● need to coordinate across clients
● use ephemeral ports	● have static IP addresses and ports
● connect as needed to make requests	● tend to have longer lifespans
● don't require dedicated hardware	● often run on dedicated machines
● requests use of these resources	● manage and control network resources

• access and use these services	• designed to provide services

7. Describe client-server architecture using an appropriate diagram.

- Clients send requests to the server over the internet
- Server runs application logic and retrieves data from database
- Server processes request, performs actions and sends response
- Clients wait for and receive responses over the network



8. How does TCP work?

In summary, TCP provides reliable in-order data transfer using error checking, sequencing, and flow control.

9. How do you write a multi-threaded server in Java?

Key classes used: ServerSocket, Socket, Thread, Runnable, ExecutorService

- Create a ServerSocket and bind it to a port
- Use ServerSocket.accept() to listen and accept connections from clients
- For each client connection, create a new thread to handle it
- In the thread, get the Socket for the connection
- Use Socket to send/receive data, such as with InputStream and OutputStream
- Access the stream and communicate with the client
- Close streams and socket when done
- Use a thread pool (ExecutorService) for managing threads
- Implement run() method in a Runnable to define thread's work
- Submit Runnables to thread pool rather than creating threads manually

10. What is an ephemeral port?

So in summary, ephemeral ports are short-lived, random client-side ports used for TCP/UDP sessions.

- They are assigned automatically from a predefined range by the OS during communication.
- They are typically in the range 1024 to 65535 on most systems.

11. Explain the following:

a. IP Address

A unique address assigned to a device on a network that uses the Internet Protocol for communication. It identifies the device and allows it to communicate with other devices on the network.

b. Protocol
A set of rules and standards that define how data is transmitted on a network. Common protocols include TCP, IP, HTTP, FTP, SMTP etc.

c. Port Number
A 16-bit number that identifies the sending and receiving application process for network communication. Common ports include 80 for HTTP, 25 for SMTP etc.

d. MAC Address
A unique identifier assigned to a network interface controller or network interface card (NIC). It is a 12-digit hexadecimal number used as a hardware identification.

e. TCP
Transmission Control Protocol. A connection-oriented protocol that provides reliable, ordered delivery of data between applications over an IP network.

f. FTP
File Transfer Protocol. Used for transferring files over a TCP/IP network. Works on client-server model and uses separate control and data connections.

g. Telnet
A protocol used for bidirectional interactive text-oriented communication over a network. Enables remote login to a host from a client.

h. SMTP
Simple Mail Transfer Protocol. Used for email transmission. Works on a client-server model. Clients send emails, server relays them.

i. POP
Post Office Protocol. A protocol used by email clients to retrieve emails from a mail server. Users connect, download messages locally, disconnect.

12. Explain about Skeleton & Stub.

Skeleton:

- Code on the server side that marshals/unmarshals parameters and return values
- Calls the actual service implementation on the server
- Sends back response to the client

Stub:

- Code on the client side that marshals call parameters
- Sends call request to server
- Waits and receives response from skeleton
- Unmarshals response and return values

13. How do we Establish a Socket Connection?

So in summary, server listens on a port, client requests connection, upon accept, streams can be opened for communication.

On the server:

- Create a ServerSocket object and bind it to a port
- Listen for connections using ServerSocket.accept()

On the client:

- Create a Socket object specifying server IP and port
- Open connection using Socket.connect()

Once connected:

- Server and client can get input and output streams
- Read and write streams to send/receive data
- Close socket when done by invoking Socket.close()

14. What are the Important methods of socket & Server classes?

Socket class:

- connect() - connects to the server
- bind() - binds socket to a local port
- getInputStream()/getOutputStream() - get input/output streams
- close() - closes the socket

ServerSocket class:

- bind() - associates the server socket with a port
- listen() - puts the server socket in listening mode
- accept() - accepts incoming client connection requests
- close() - closes the server socket

15. Why do we close created connections before the system Shutting Down?

- So in essence, we close connections properly to release resources in a defined sequence and prevent any unwanted side effects.

- To preventOverflow Errors

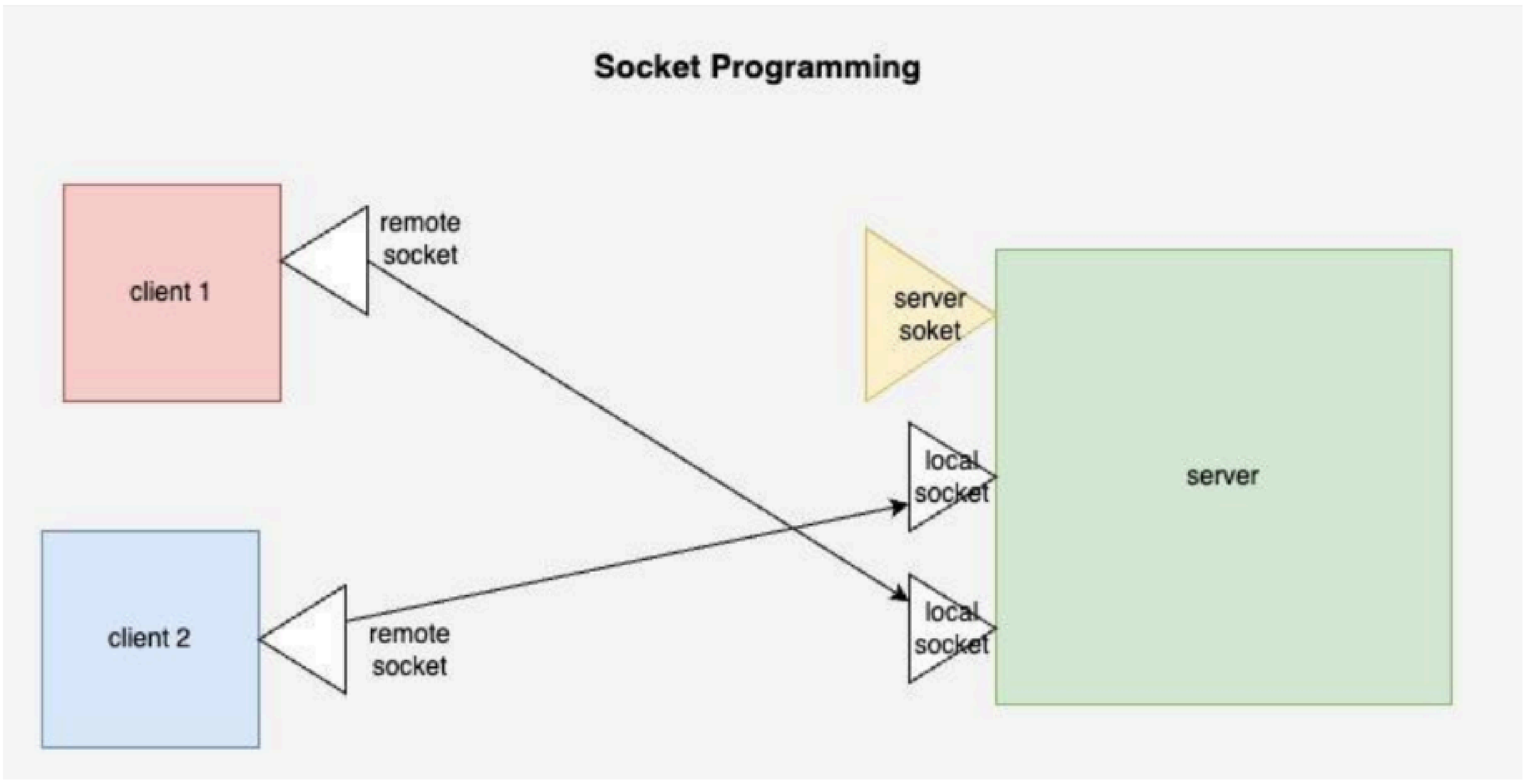
- To enable clean restarts

- For faster connection re-establishment

##

1. Socket Programming

Socket Programming is a way of connecting two nodes on a network to communicate with each other.



2. Endpoint

An endpoint is a combination of an IP address and a port number.

3.Client Side Programming

- In the case of client-side programming, the client will first wait for the server to start.
- Once the server is up and running, it will send the requests to the server. After that, the client will wait for the response from the server.

  Steps to initiate a Client Request,

  1. Establishing a connection.
  2. Communication.
  3. Closing the connection.

4. Data Stream

A stream is a sequence of data. In Java, a stream is composed of bytes. It's called a stream because it is like a stream of water that continues to flow.

5. Streams used In Socket Programming.

1. InputStream
    - Java application uses an input stream to read data from a source; it may be a file, an array, peripheral device or socket.

2. OutputStream
    - Java application uses an output stream to write data to a destination; it may be a file, an array, peripheral device or socket.

6.Closing the Connection

- The socket connection is closed explicitly once the message to the server is sent.

- It is important to close the connection by closing the socket as well as input/output streams once everything is done.

- Java program to implement socket connection at client side is to be coded......

7.Server Side Programming

- Basically, the server will instantiate its object and wait for the client request.
- Once the client sends the request, the server will communicate back with the response.

8.Network
A computer collection. E kiyanne computers godak ekathu karala hadana jalayak…
    Eg:network devices , computer

9.Server
Processor godak ekathu karala hadana ekaki….

10.ISP (Internet service provider)
Meka haraha thama internet ekata connect wenne….

11.DMS (Domain Main System)
Meken thama human rederble system ekak Ip Address ekakata convert karanne….

12.Data bracket
Meka data unit ekaki…network athule data ekak transfer karanna thama meka ganne…

13.firewall
Meka security system ekaki…meka incoming and outgoing network athara security eka maintain karanawa…e kiyanne unworther traffic disable karanawa…

14.Threads
Meka execution single unit ekaki…Threads kiyala class ekakuth thiyenawa…Threads interface implement karannath puluwan..

15.Thread pool

Meke threads thama manage karanne…

16.Process base

Meke theruma thama eka computer ekak athule multiple execution karanawa…