School of
Computer Studies

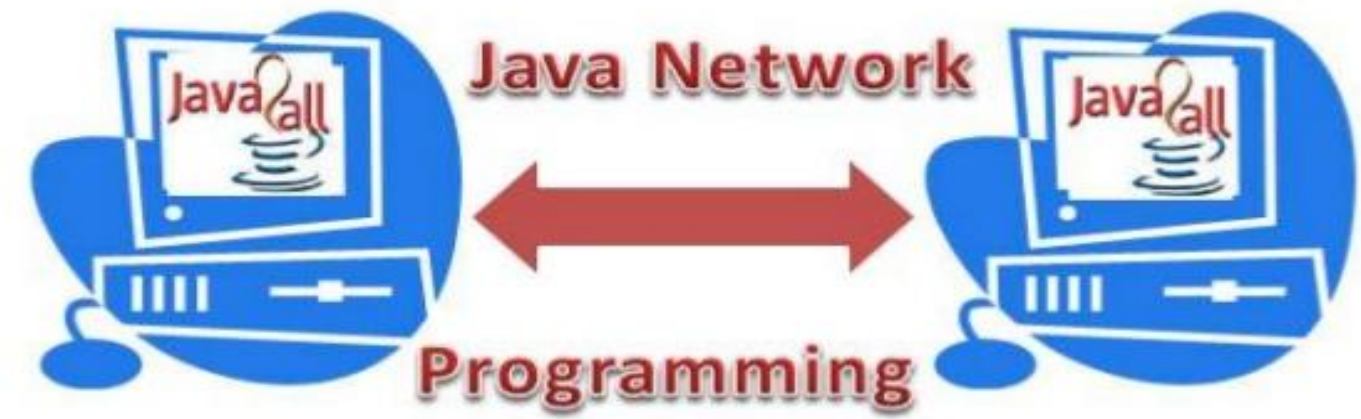
AHMEDABAD UNIVERSITY

iMCA Semester-V

Advanced Java Technologies (AJT)

Java Networking

Aditya Patel

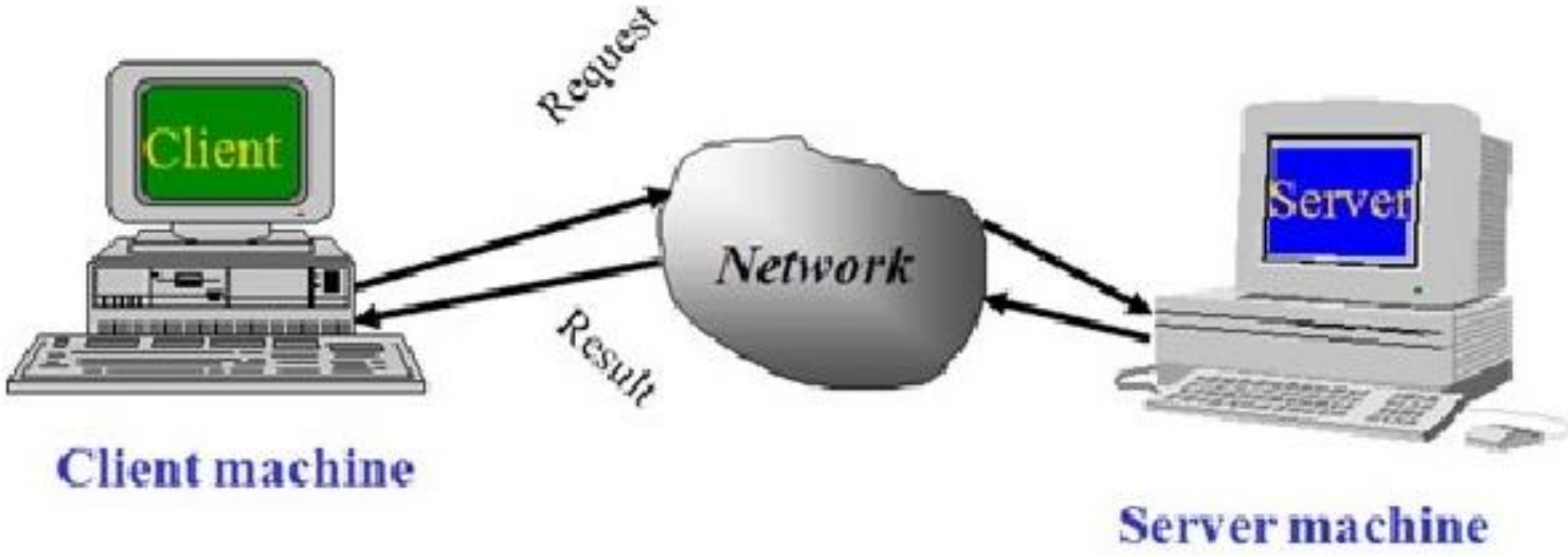


AHMEDABAD UNIVERSITY

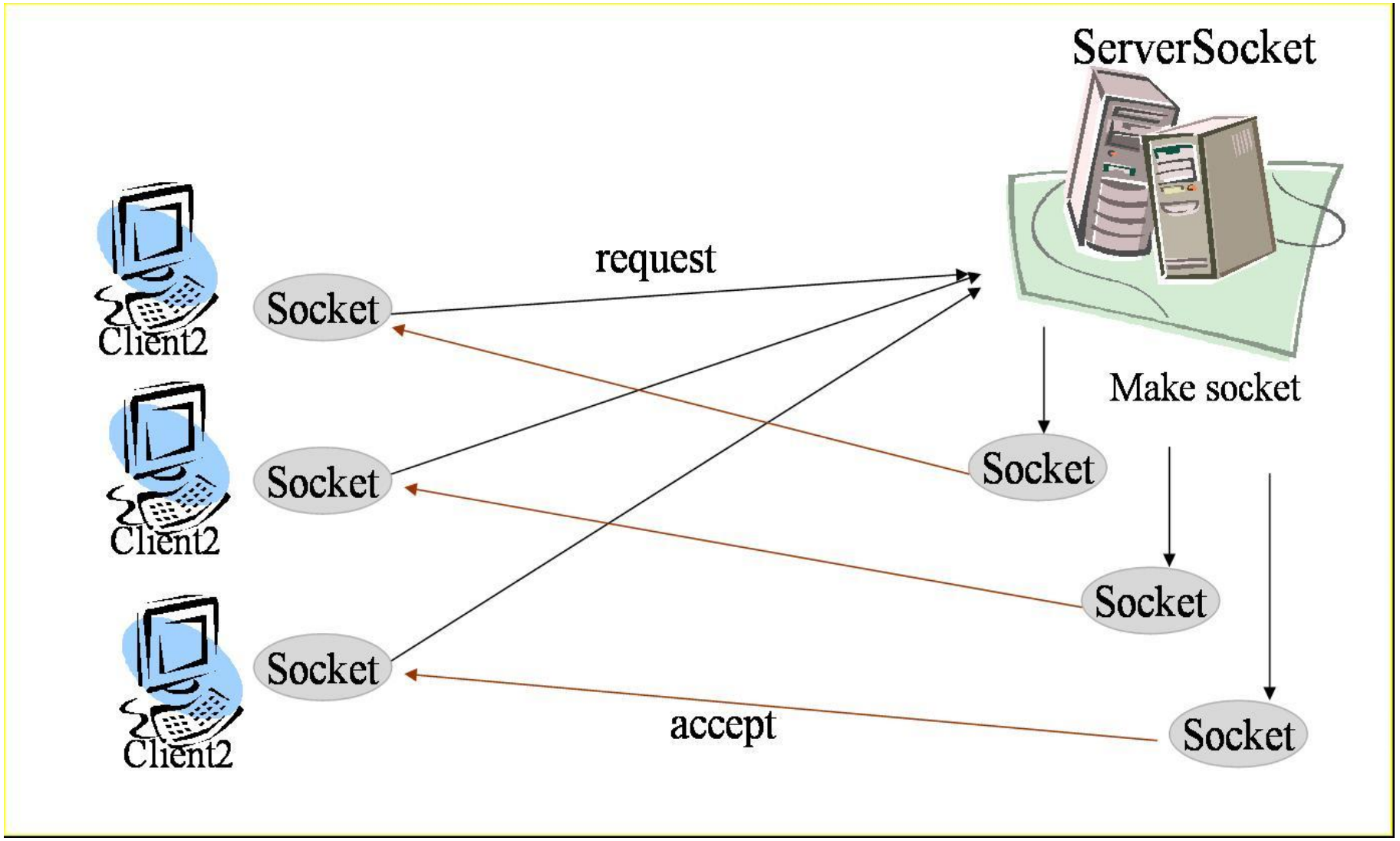
Java Networking

- Networking is a concept of connecting two or more computing devices together so that we can share data and resources.
- Network/Socket programming refers to writing programs that are able to perform I/O – read/write over the network
- Programs able to communicate to remote servers over the network using network protocols like TCP or UDP.

Java Networking



Java Networking



Basic Network Programming Concepts

- IP Address - An IP (Internet Protocol) address is a number that uniquely identifies a computer on a network.
- This are 32-bit numbers E.g. 192.168.1.1
- → Refer IPDemo.java

- Port No – logical concept - is a way to identify/communicate with a specific process (out of many) on a computer
- Program/service listens for incoming packets of data from network – on specific port no.
- There are o to 1023 well-known port numbers

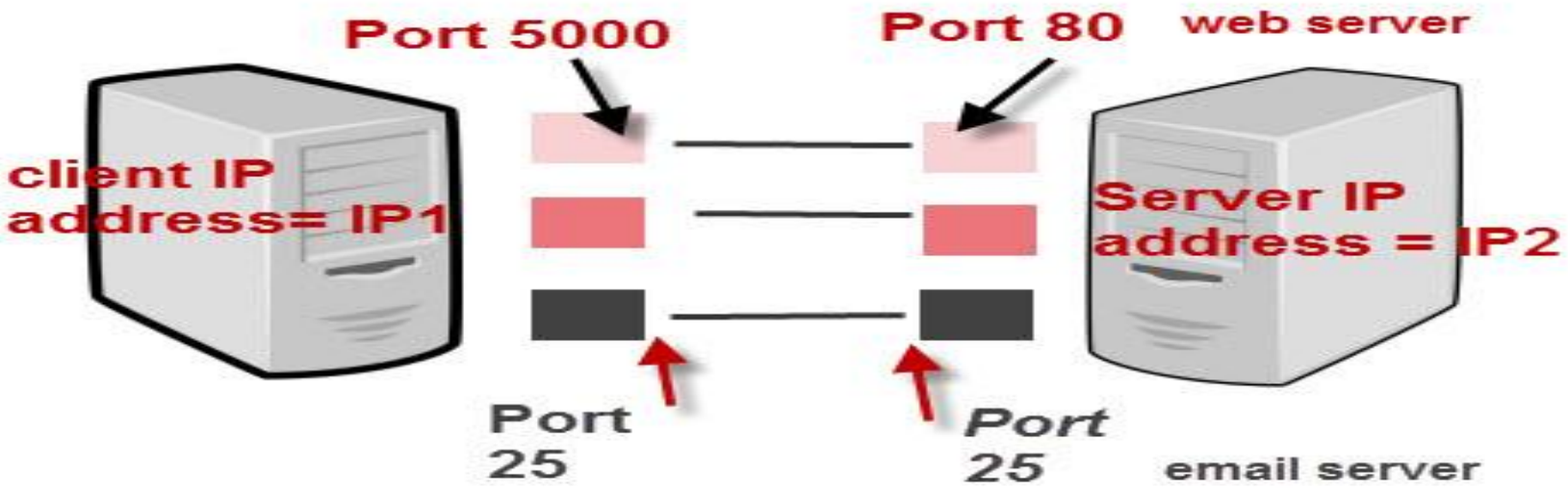
AHMEDABAD UNIVERSITY

Basic Network Programming Concepts

- Well known port nos – HTTP/Web Server listens on port 80
- 21: File Transfer Protocol (FTP)
- 22: Secure Shell (SSH)
- 23: Telnet remote login service
- 25: Simple Mail Transfer Protocol (SMTP)
- 80: Hypertext Transfer Protocol (HTTP) used in the World Wide Web

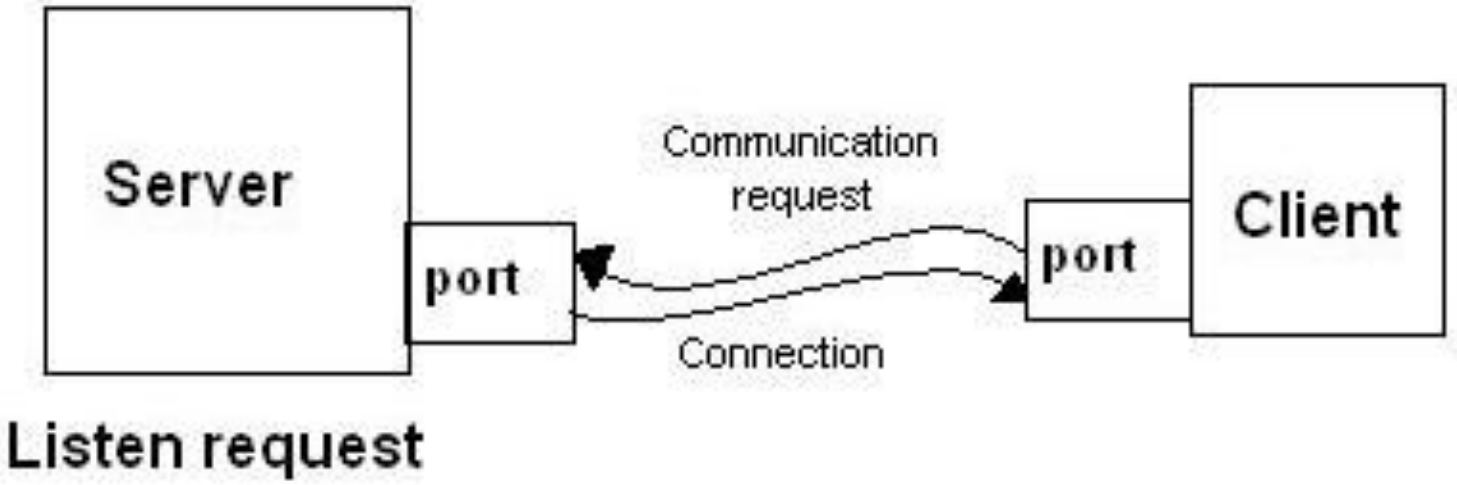
Basic Network Programming Concepts

- **Socket - Combination of IP address + port no.**
- Socket is one endpoint of a two-way communication link between two programs running on the network.



Basic Network Programming Concepts

- Socket - Combination of IP address + port no.
- Socket is one endpoint of a two-way communication link between two programs running on the network.



Java Socket Programming - Server

- java.net - Simple API/programming model
- To create a server – use ServerSocket
- java.net.ServerSocket class provides a mechanism for the server program to listen for clients and establish connections with them.
- ServerSocket ss=new ServerSocket(6666);
- Socket s=ss.accept();
- //wait for incoming connection from client and establishes connection

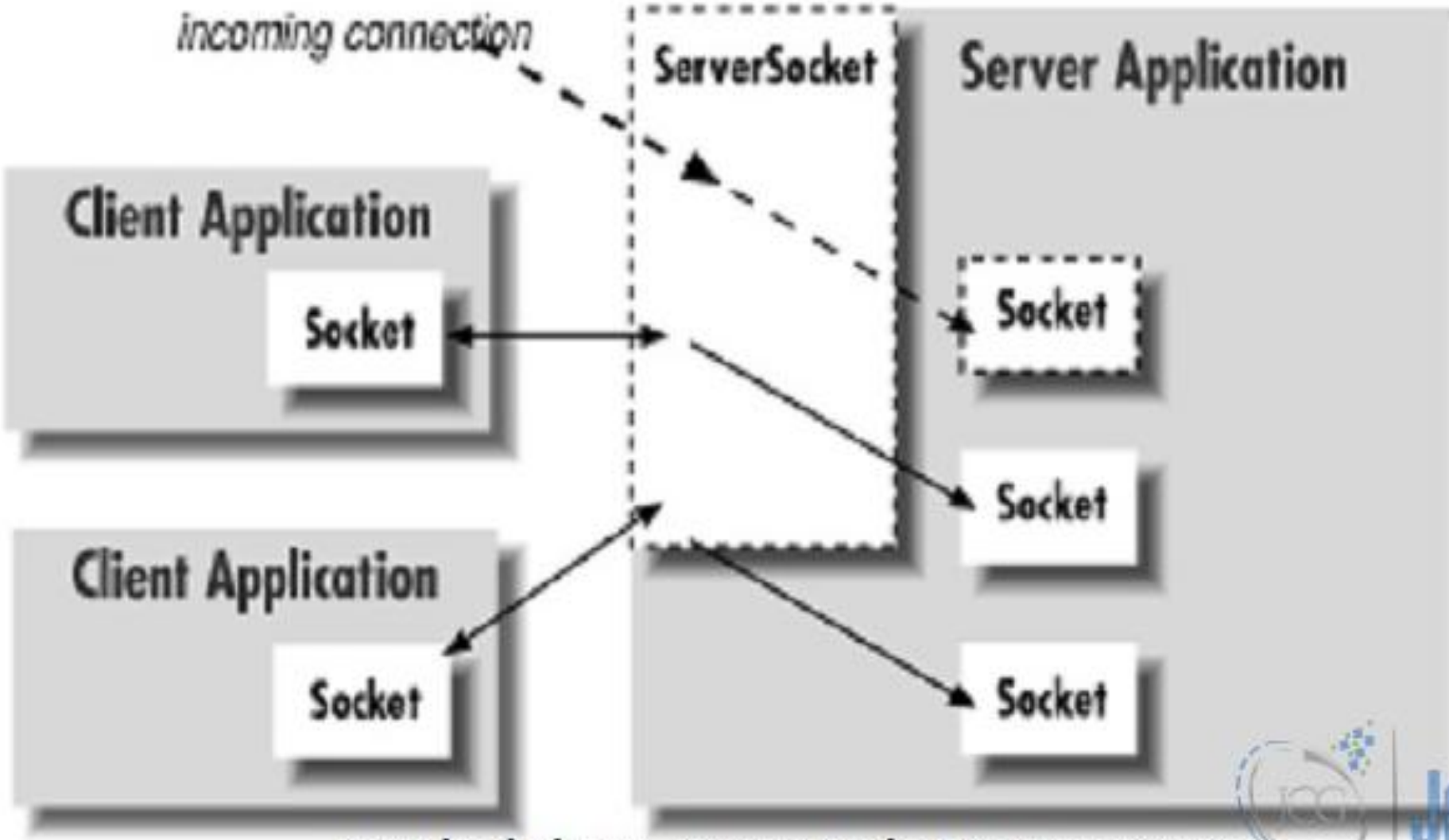
AHMEDABAD UNIVERSITY

Java Socket Programming - Client

- To create a client – use java.net.Socket class
- Client sends a connection request to server specifying its IP and portno
- IP or hostname are allowed
- Socket s=new Socket("localhost",6666);

- Start writing/reading data to socket
- DataOutputStream dout=new DataOutputStream(s.getOutputStream());
- dout.writeUTF("Hello Server");

Socket Programming



Standard Client - Server Socket Communication

Socket Programming

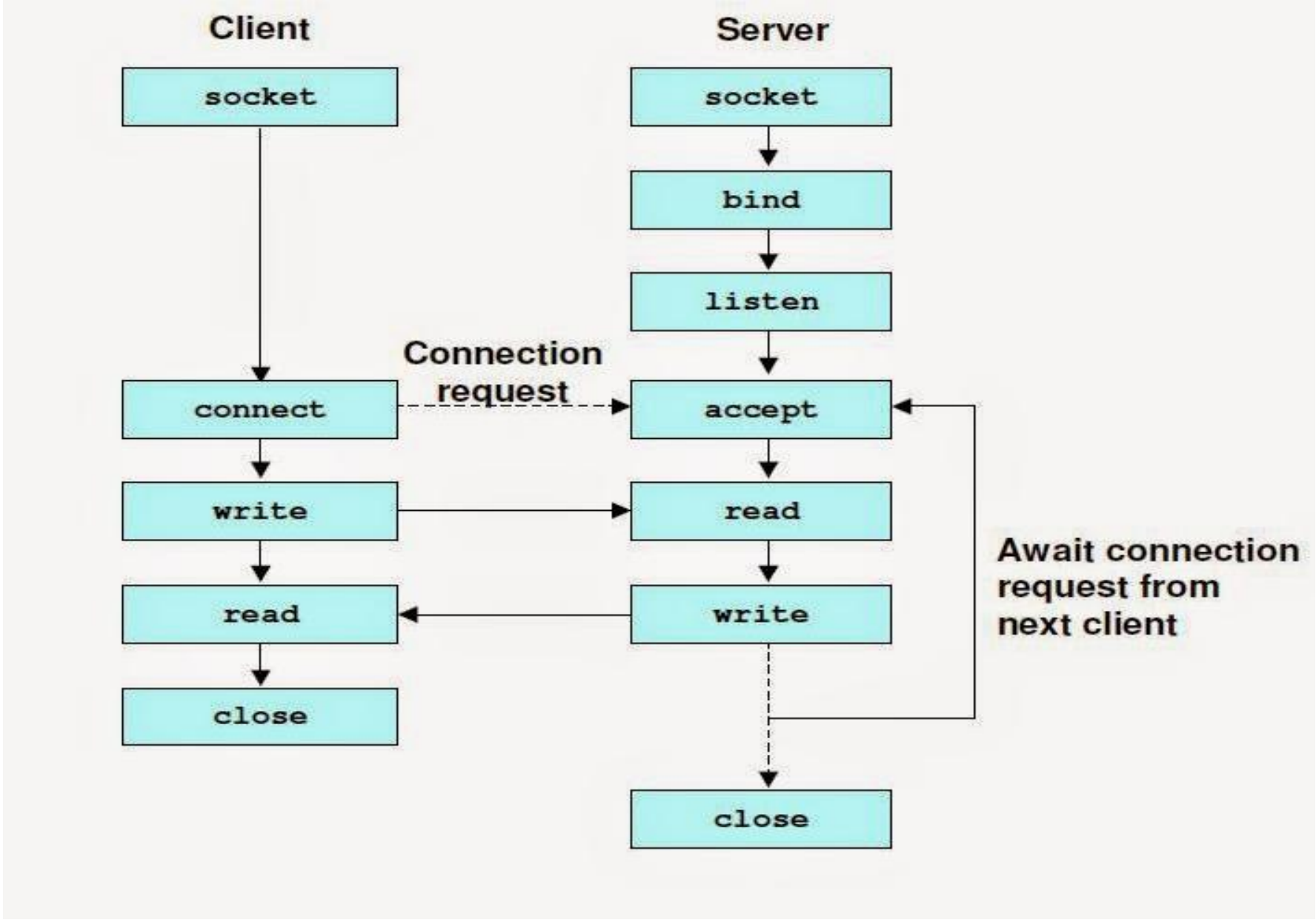
TCP Server Socket

- **Step-1** Create **`ServerSocket`** Object
- **Step-2** Wait for the client request
- **Step-3** Create I/O streams for communication to the client.
- **Step-4** Perform communication with client
- **Step-5** Close Socket

TCP Client Socket

- **Step-1** Create **`Socket`** Object
- **Step-2** Create I/O streams for communication with the server
- **Step-3** Perform I/O or communication with the server
- **Step-4** Close Socket

Socket Programming



AHMEDABAD UNIVERSITY

Java Networking

- This is used for mainly in two things :
  - **Socket Programming**
  - **URL Processing**

Socket Programming

The following steps occur when establishing a TCP connection between two computers using sockets

1. The server instantiates a ServerSocket object, denoting which port number communication is to occur on.
2. The server invokes the accept() method of the ServerSocket class. This method waits until a client connects to the server on the given port.

Socket Programming

3. After the server is waiting, a client instantiates a Socket object, specifying the server name and the port number to connect to.
4. The constructor of the Socket class attempts to connect the client to the specified server and the port number. If communication is established, the client now has a Socket object capable of communicating with the server.
5. On the server side, the accept() method returns a reference to a new socket on the server that is connected to the client's socket.

ServerSocket Class Methods

The **java.net.ServerSocket** class is used by server applications to obtain a port and listen for client requests.

The ServerSocket class has four constructors –

1. **public ServerSocket(int port) throws IOException**

Attempts to create a server socket bound to the specified port. An exception occurs if the port is already bound by another application

2. **public ServerSocket(int port, int backlog) throws IOException**

Similar to the previous constructor, the backlog parameter specifies how many incoming clients to store in a wait queue.

AHMEDABAD UNIVERSITY

The ServerSocket class

1. **public ServerSocket(int port, int backlog, InetAddress address) throws IOException**
   Similar to the previous constructor, the InetAddress parameter specifies the local IP address to bind to The InetAddress is used for servers that may have multiple IP addresses.
1. 4. **public ServerSocket() throws IOException**
2. Creates an unbound server socket. When using this constructor, use the bind() method when you are ready to bind the server socket.

AHMEDABAD UNIVERSITY

some of the common methods of the ServerSocket class –

1. **public int getLocalPort()**

Returns the port that the server socket is listening on. This method is useful if you passed in 0 as the port number in a constructor and let the server find a port for you.

2. **public Socket accept() throws IOException**

Waits for an incoming client. This method blocks until either a client connects to the server on the specified port or the socket times out, assuming that the time-out value has been set using the setSoTimeout() method. Otherwise, this method blocks indefinitely.

some of the common methods of the ServerSocket class –

3. **public void setSoTimeout(int timeout)**
Sets the time-out value for how long the server socket waits for a client during the accept()

4. **public void bind(SocketAddress host, int backlog)**
Binds the socket to the specified server and port in the SocketAddress object. Use this method if you have instantiated the ServerSocket using the no-argument constructor.

AHMEDABAD UNIVERSITY

Socket Class Methods

The **java.net.Socket** class represents the socket that both the client and the server use to communicate with each other. The client obtains a Socket object by instantiating one, whereas the server obtains a Socket object from the return value of the accept() method.

java.net.Socket

1. **public Socket(String host, int port) throws UnknownHostException, IOException.**

This method attempts to connect to the specified server at the specified port. If this constructor does not throw an exception, the connection is successful and the client is connected to the server.

2. **public Socket(InetAddress host, int port) throws IOException**

This method is identical to the previous constructor, except that the host is denoted by an InetAddress object.

Socket Client Example

The following GreetingClient is a client program that connects to a server by using a socket and sends a greeting, and then waits for a response.

Socket Server Example

The following GreetingServer program is an example of a server application that uses the Socket class to listen for clients on a port number specified by a command-line argument −

Socket Programming

The following steps occur when establishing a TCP connection between two computers using sockets

1. The server instantiates a ServerSocket object, denoting which port number communication is to occur on.
2. The server invokes the accept() method of the ServerSocket class. This method waits until a client connects to the server on the given port.

AHMEDABAD UNIVERSITY

Java Networking

The java.net package support 2 network protocols −

- **TCP** − TCP stands for Transmission Control Protocol, which allows for reliable connection-oriented communication between 2 apps. TCP is typically used over the Internet Protocol, which is referred to as TCP/IP.

- **UDP** − UDP stands for User Datagram Protocol, a connection-less protocol that allows for packets of data to be transmitted between applications.

Socket Programming

TCP	UDP
Reliable	Unreliable
Connection-oriented	Connectionless
Segment retransmission and flow control through windowing	No windowing or retransmission
Segment sequencing	No sequencing
Acknowledge segments	No acknowledgement