دانشکده مهندسی و علوم کامپیوتر

برنامه نویسی پیشرفته

وحیدی اصل

برنامه نویسی شبکه- بخش اول

Java Networking

- Java Networking is a concept of connecting two or more computing devices together so that we can share resources.
- Java socket programming provides facility to share data between different computing devices.
- Advantage of Java Networking
  - sharing resources
  - centralize software management

Do you know?

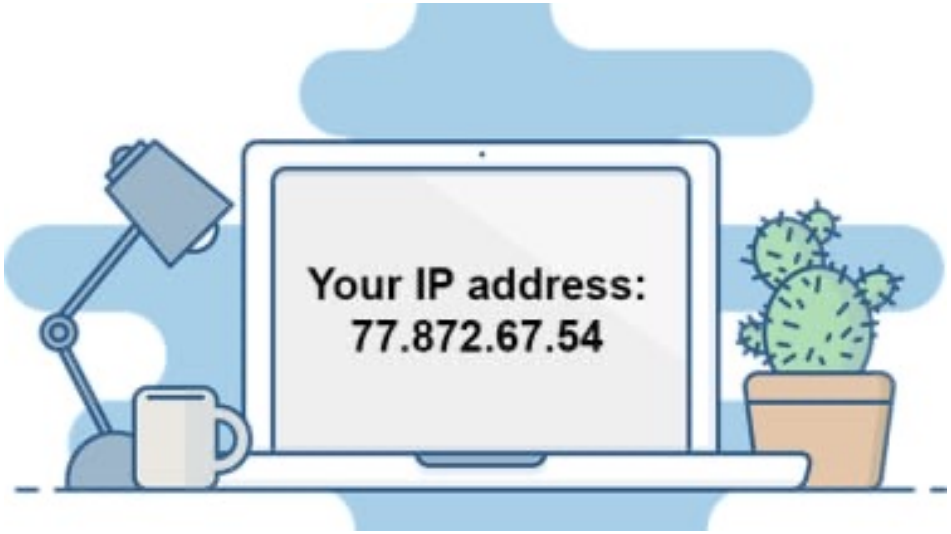
- How to perform connection-oriented Socket Programming in networking ?
- How to display the data of any online web page ?
- How to get the IP address of any host name e.g. www.google.com ?
- How to perform connection-less socket programming in networking ?

Java Networking Terminology

- IP Address
- Protocol
- Port Number
- MAC Address
- Connection-oriented and connection-less protocol
- Socket

IP Address

- IP address is a unique number assigned to a node of a network e.g. 192.168.0.1 .
- It is composed of octets that range from 0 to 255.
- It is a logical address that can be changed.



Protocol

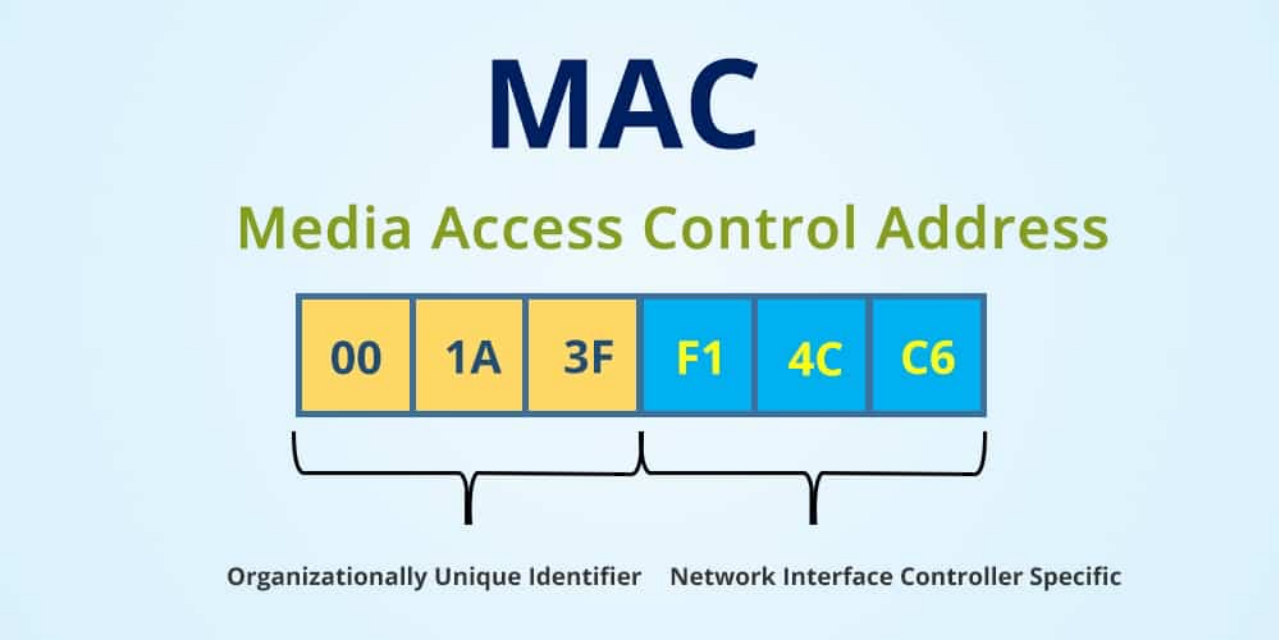
- A protocol is a set of rules basically that is followed for communication. For example:
  - TCP: Transmission Control Protocol is a standard that defines how to establish and maintain a network conversation through which application programs can exchange data. **TCP** works with the Internet Protocol (IP), which defines how computers send packets of data to each other.
  - UDP: User Datagram Protocol (UDP) is a Transport Layer protocol. UDP is a part of Internet Protocol suite, referred as UDP/IP suite. Unlike TCP, it is unreliable and connectionless protocol. So, there is no need to establish connection prior to data transfer.
  - FTP: The File Transfer Protocol (FTP) is a standard network protocol used for the transfer of computer files between a client and server on a computer network.
  - Telnet:a type of client-server protocol that can be used to open a command line on a remote computer, typically a server.
  - SMTP: The Simple Mail Transfer Protocol (SMTP) is a communication protocol for electronic mail transmission.
  - POP : Post Office Protocol (POP) is a type of computer networking and Internet standard protocol that extracts and retrieves email from a remote mail server for access by the host machine

Port Number

- The port number is used to uniquely identify different applications.
- It acts as a communication endpoint between applications.
- The port number is associated with the IP address for communication between two applications.

MAC Address

- MAC (Media Access Control) Address is a unique identifier of NIC (Network Interface Controller).
- A network node can have multiple NIC but each with unique MAC.



Connection-oriented and connection-less protocol

- In connection-oriented protocol, acknowledgement is sent by the receiver.
  - So it is reliable but slow. The example of connection-oriented protocol is TCP.
- But, in connection-less protocol, acknowledgement is not sent by the receiver.
  - So it is not reliable but fast. The example of connection-less protocol is UDP.

java.net package

- The java.net package provides many classes to deal with networking applications in Java.

Authenticator	InetSocketAddress	ResponseCache	URLDecoder
CacheRequest	InetAddress	SecureCacheResponse	URLEncoder
CacheResponse	Inet4Address	ServerSocket	URLStreamHandler
ContentHandler	Inet6Address	Socket	
CookieHandler	IDN	SocketAddress	
CookieManager	HttpURLConnection	SocketImpl	
DatagramPacket	HttpCookie	SocketPermission	
DatagramSocket	NetPermission	StandardSocketOptions	
DatagramSocketImpl	NetworkInterface	URI	
InterfaceAddress	PasswordAuthentication	URL	
JarURLConnection	Proxy	URLClassLoader	
MulticastSocket	ProxySelector	URLConnection	

What we will learn in Networking

- Networking and Networking Terminology
- Socket Programming (Connection-oriented)
- URL class
- Displaying data of a webpage by URLConnection class
- InetAddress class
- DatagramSocket and DatagramPacket (Connection - less)

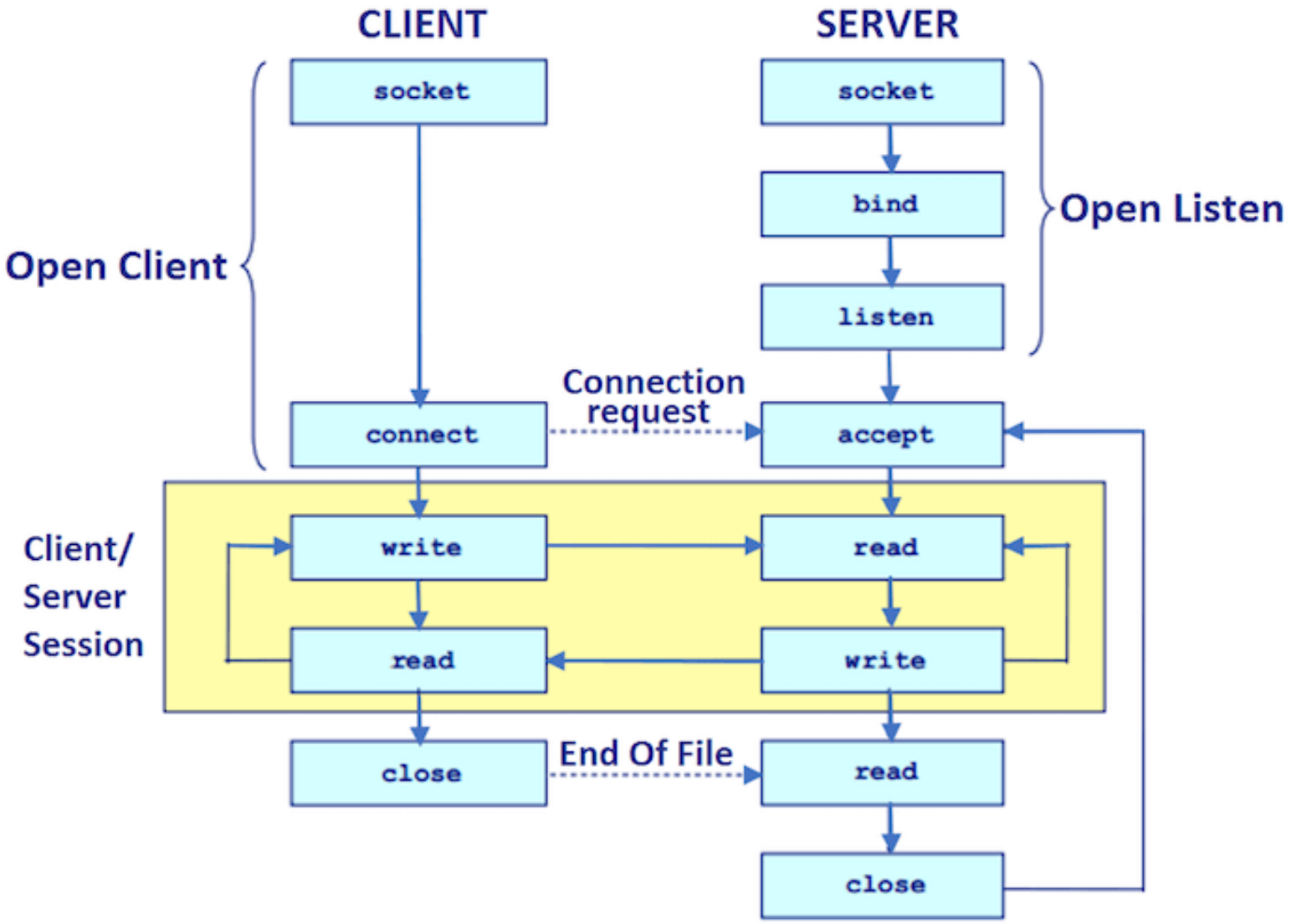
Java Socket Programming

- Java Socket programming is used for communication between the applications running on different JRE.
- Java Socket programming can be connection-oriented or connection-less.
- Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

Java Socket Programming

- The client in socket programming must know two information:
  - IP Address of Server, and
  - Port number.

نمایی از برنامه نویسی سوکت در حالت connection-oriented



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
- If the ServerSocket constructor does not throw an exception, it means that your application has successfully bound to the specified port and is ready for client requests.

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.
3	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, allowing the server to specify which of its IP addresses to accept client requests on.
4	public ServerSocket() throws IOException Creates an unbound server socket. When using this constructor, use the bind() method when you are ready to bind the server socket.

ServerSocket Class Methods

- In very basic one-way Client and Server setup:
  - a Client connects, sends messages to server and the server shows them using socket connection.
  - There's a lot of low-level stuff that needs to happen for these things to work but the Java API networking package (java.net) takes care of all of that, making network programming very easy for programmers.
- To connect to other machine we need a socket connection.
- A socket connection means the two machines have information about each other's network location (IP Address) and TCP port. The java.net.Socket class represents a Socket.

```
Socket socket = new Socket("127.0.0.1", 6666);
```

- **First argument** – IP address of Server. (127.0.0.1 is the IP address of localhost, where code will run on single stand-alone machine).
- **Second argument** – TCP Port. (Just a number representing which application to run on a server. For example, HTTP runs on port 80. Port number can be from 0 to 65535)

- **Creating Server:**
  - To create the server application, we need to create the instance of ServerSocket class.
  - Here, we are using 6666 port number for the communication between the client and server. You may also choose any other port number.
  - The accept() method waits for the client. If clients connects with the given port number, it returns an instance of Socket. The accept() method blocks (just sits there) until a client connects to the server.

```
ServerSocket ss=new ServerSocket(6666);
Socket s=ss.accept();//establishes connection and waits for the client
```

- **Creating Client:**
  - To create the client application, we need to create the instance of Socket class.
  - Here, we need to pass the IP address or hostname of the Server and a port number. Here, we are using "localhost" because our server is running on same system.

127.0.0.1

```
Socket s=new Socket("localhost",6666);
```

Let's see a simple of Java socket programming where client sends a text and server receives and prints it.

- **Communication**
  - To communicate over a socket connection, streams are used to both input and output the data.

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

- **Establish a Socket Connection**
  - To write a server application two sockets are needed.
  - A ServerSocket which waits for the client requests (when a client makes a new Socket())
  - A plain Socket socket to use for communication with the client.

- **Communication**
  - getOutputStream() method is used to send the output through the socket.
- **Close the Connection**
  - After finishing, it is important to close the connection by closing the socket as well as input/output streams.

- To run the Client and Server application on your machine, compile both of them. Then first run the server application and then run the Client application.

مثالی از کد سرور

```
import java.io.*;
import java.net.*;
public class MyServer {
public static void main(String[] args){
try{
ServerSocket ss=new ServerSocket(6666);
Socket s=ss.accept();//establishes connection
DataInputStream dis=new DataInputStream(s.getInputStream());
String  str=(String)dis.readUTF();
System.out.println("message= "+str);
ss.close();
}catch(Exception e){System.out.println(e);}
}
}
```

مثالی از کد کلاینت

```
import java.io.*;
import java.net.*;
public class MyClient {
public static void main(String[] args) {
try{
Socket s=new Socket("localhost",6666);
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
dout.writeUTF("Hello Server");
dout.flush();
dout.close();
s.close();
}catch(Exception e){System.out.println(e);}
}
}
```

Example of Java Socket Programming (Read-Write both side)

- In this example, client will write first to the server then server will receive and print the text.
- Then server will write to the client and client will receive and print the text.
- The step goes on.

MyServer code

```
import java.net.*;
import java.io.*;
class MyServer{
public static void main(String args[])throws Exception{
ServerSocket ss=new ServerSocket(3333);
Socket s=ss.accept();
DataInputStream din=new DataInputStream(s.getInputStream());
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String str="",str2="";
while(!str.equals("stop")){
str=din.readUTF();
System.out.println("client says: "+str);
str2=br.readLine();
dout.writeUTF(str2);
dout.flush();
}
din.close();
s.close();
ss.close();
}}
```

MyClient code

```
import java.net.*;
import java.io.*;
class MyClient{
public static void main(String args[])throws Exception{
Socket s=new Socket("localhost",3333);
DataInputStream din=new DataInputStream(s.getInputStream());
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String str="",str2="";
while(!str.equals("stop")){
str=br.readLine();
dout.writeUTF(str);
dout.flush();
str2=din.readUTF();
System.out.println("Server says: "+str2);
}

dout.close();
s.close();
}}
```