Socket Programming with Java

By

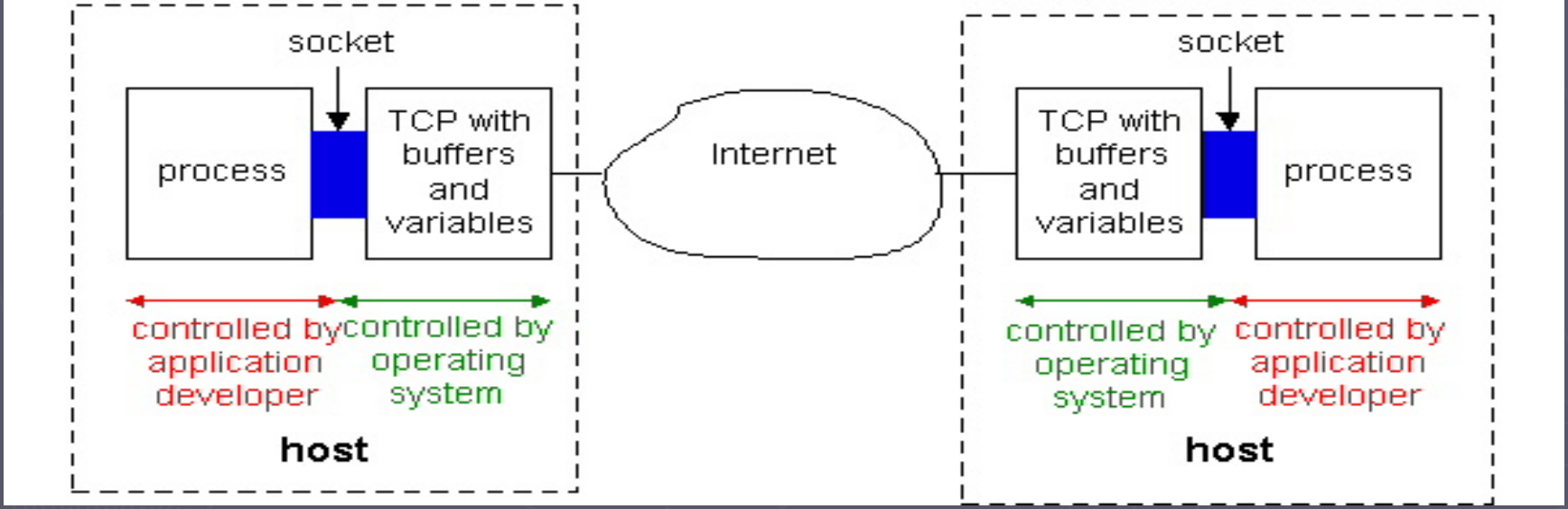
Sunirmal Khatua

University of Calcutta

What is a Socket?

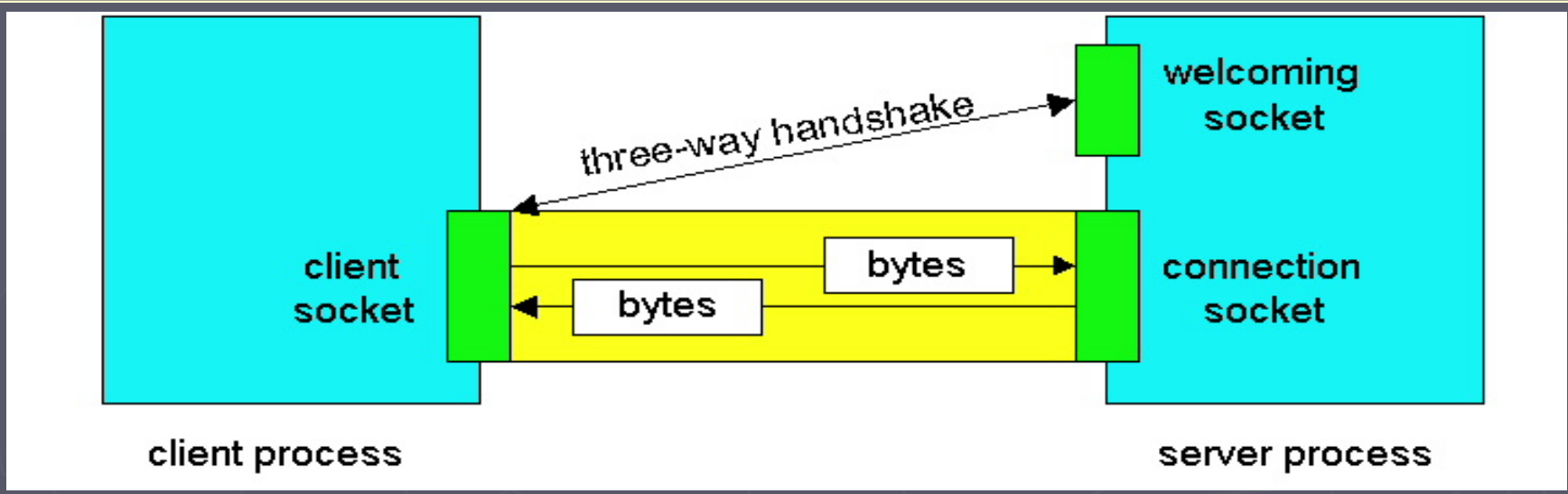
- A logical End Point of a communication, a combination of (IP & Port)
- Types:
  - Stream Socket
    - ✔ Reliable two-way connected communication.
    - ✔ Used By TCP
  - Datagram Socket
    - ✔ Unreliable connectionless communication.
    - ✔ Used By UDP
- Socket Pair:
  - Specified the two endpoints that uniquely identifies each TCP connection.

Socket Programming with TCP

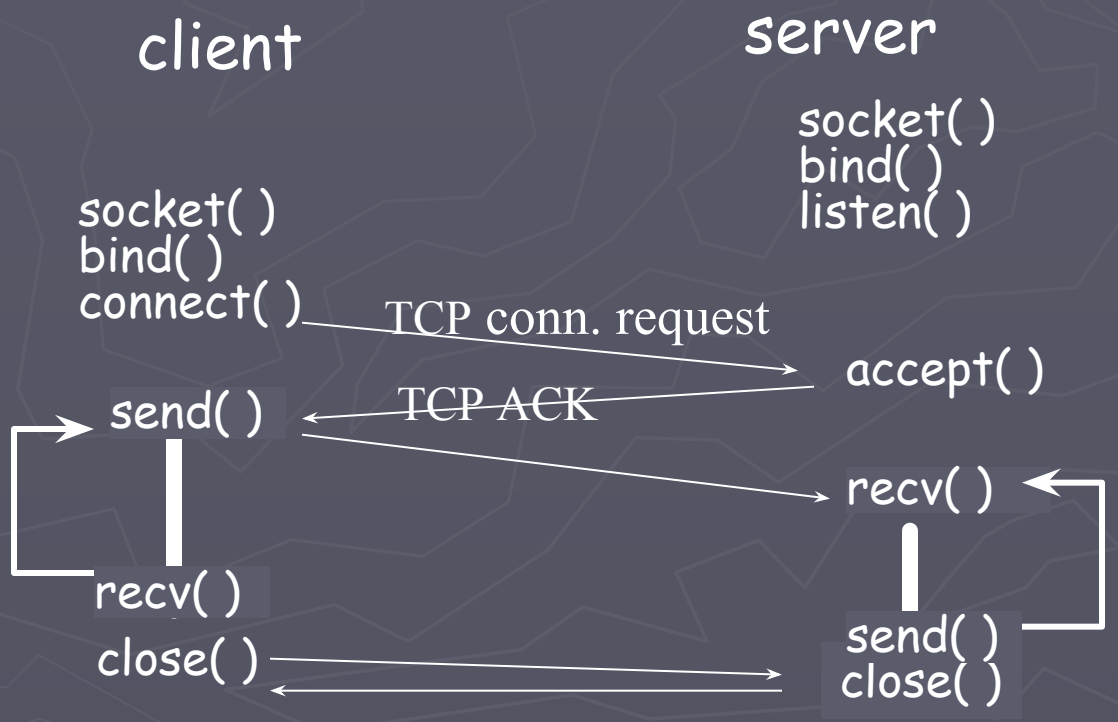


- Functional Primitives:
  - socket () : Create a socket
  - bind() : bind a socket to a local IP address and port #
  - listen() : passively waiting for connections
  - connect() : initiating connection to another socket
  - accept() : accept a new connection
  - Write() : write data to a socket
  - Read() : read data from a socket
  - sendto() : send a datagram to another UDP socket
  - recvfrom() : read a datagram from a UDP socket
  - close() : close a socket (tear down the connection)

Server & Client Sockets



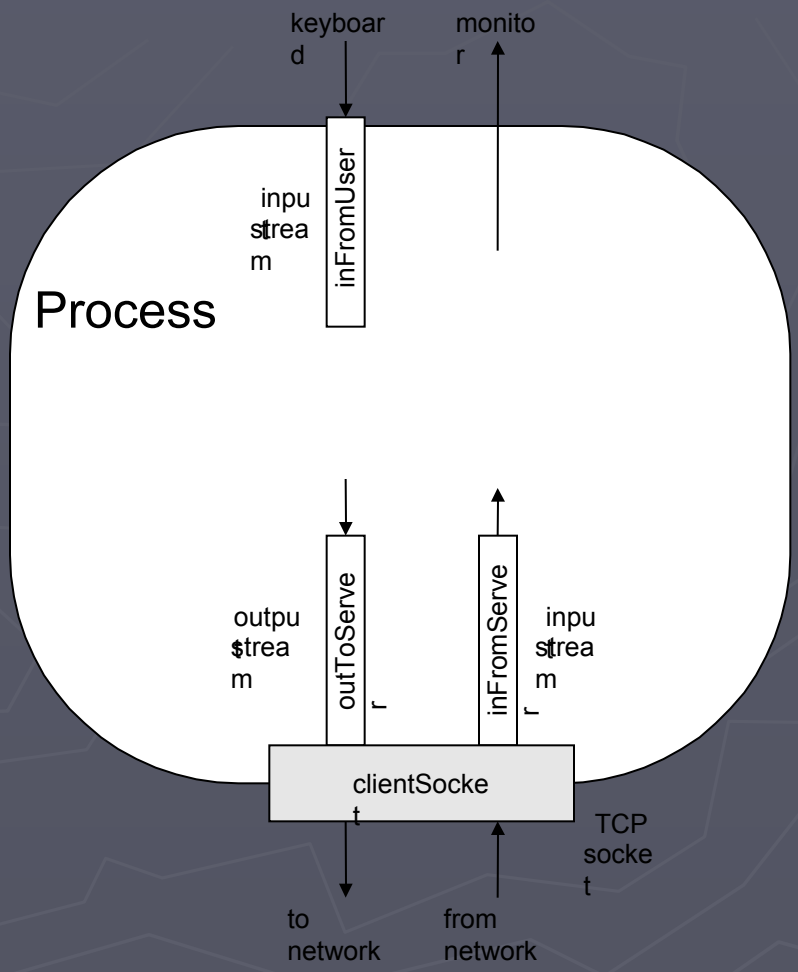
- Server:
  - Welcoming Socket
  - Connection Socket
- Client:
  - Client Socket



Java TCP Sockets

- Package *java.net*
  - Class *java.net.Socket*
    - ✔ Implements Client Sockets & Connection Sockets
    - ✔ Constructors & Methods:
      - *Socket(String host, int port)*
      - *InputStream getInputStream();*
      - *OutputStream getOutputStream();*
      - *close();*
  - Class *java.net.ServerSocket*
    - ✔ Implements Welcome Sockets
    - ✔ Constructors & Methods:
      - *ServerSocket(int port)*
      - *Socket accept()*

Communication Buffers



TCPServer.java

```
public class TCPServer {
    public static void main(String args[]) throws Exception{
     String sentence;
     String modifiedSentence;
     ServerSocket welcomeSocket = new ServerSocket(port);
     while(true){
          Socket connectionSocket = welcomeSocket.accept();
               Scanner inFromClient = new Scanner(connectionSocket.getInputStream());
          OutputStream outToClient = connectionSocket.getOutputStream();
          sentence = inFromClient.readLine();
               modifiedSentence = sentence.toUpperCase();
               outToClient.write((modifiedSentence+"\n").getBytes());
               connectionSocket.close();
     }
    }
}
```

TCPClient.java

```
public class TCPClient {
    public static void main(String args[]) throws Exception{
      String sentence;
      String modifiedSentence;
      Scanner inFromUser = new Scanner(System.in);
      sentence = inFromUser.readLine();
      Socket clientSocket = new Socket("IP Address",port);
      OutputStream outToServer = clientSocket.getOutputStream();
            Scanner inFromServer = new Scanner(clientSocket.getInputStream());
      outToServer.write((sentence+"\n").getBytes());
      modifiedSentence = inFromServer.readLine();
      System.out.println("FROM SERVER : "+modifiedSentence);
      clientSocket.close();
    }
}
```

Sockets Programming with UDP

- Connectionless & Unreliable
- No need of welcoming socket
- No streams are attached to the socket
- Sender need to create packets with IP & port of the destination. The sender's IP & port are automatically attached to each Packet
- Receiver need to retrieve the sender's IP & port to return back to the Sender.

Java UDP Sockets

- Package *java.net*
  - Class *java.net.DatagramSocket*
    - ✔ Used for sending & receiving datagram packets.
    - ✔ Constructors & Methods:
      - *DatagramSocket()*
      - *DatagramSocket(int port)*
      - *void receive(DatagramPacket p);*
      - *void send(DatagramPacket p);*
      - *close();*
  - Class *java.net.DatagramPacket*
    - ✔ Constructors & Methods:
      - *DatagramPacket(byte[] data, long size)*
      - *DatagramPacket(byte[] data, long size, String Ip, int port)*
      - *getAddress()*
      - *getPort()*
      - *getData()*

UDPServer.java

```
public class UDPServer {
    public static void main(String args[]) throws Exception{
      byte[] receiveData = new byte[1024];
      byte[] sendData = new byte[1024];
      DatagramSocket serverSocket = new DatagramSocket(port);
      while(true){
            DatagramPacket receivePacket =
                  new DatagramPacket(receiveData, receiveData.length);
            serverSocket.receive(receivePacket);
            String sentence = new String(receivePacket.getData());
            InetAddress IPAddress = receivePacket.getAddress();
            int port = receivePacket.getPort();
            String modifiedSentence = sentence.toUpperCase();
            sendData = modifiedSentence.getBytes();
            DatagramPacket sendPacket =
                  new DatagramPacket(sendData, sendData.length, IPAddress, port);
            serverSocket.send(sendPacket);
      }
    }
}
```

UDPClient.java

```
public class UDPClient {
    public static void main(String args[]) throws Exception{
      byte[] sendData = new byte[1024];
      byte[] receiveData = new byte[1024];
      BufferedReader inFromUser = new BufferedReader(
                                  new InputStreamReader(System.in));
      String sentence = inFromUser.readLine();
      DatagramSocket clientSocket = new DatagramSocket();
      InetAddress IPAddress = InetAddress.getByName("HostName");
      sendData = sentence.getBytes();
      DatagramPacket sendPacket =
                  new DatagramPacket(sendData, sendData.length, IPAddress, port);
      clientSocket.send(sendPacket);
      DatagramPacket receivePacket =
                  new DatagramPacket(receiveData, receiveData.length);
      clientSocket.receive(receivePacket);
      String modifiedSentence = new String(receivePacket.getData());
      System.out.println("FROM SERVER:" + modifiedSentence);
      clientSocket.close();
    }
}
```