

Introduction to Socket Programming

Abdelghani Benharref
Ph.D. Candidate, ECE

October 5th, 2007

Outline

- n Introduction to sockets
 - n Basic concepts
 - n Connection-oriented
 - n Connectionless
- n Generic client/server examples
 - n C
 - n Java
- n Conclusion

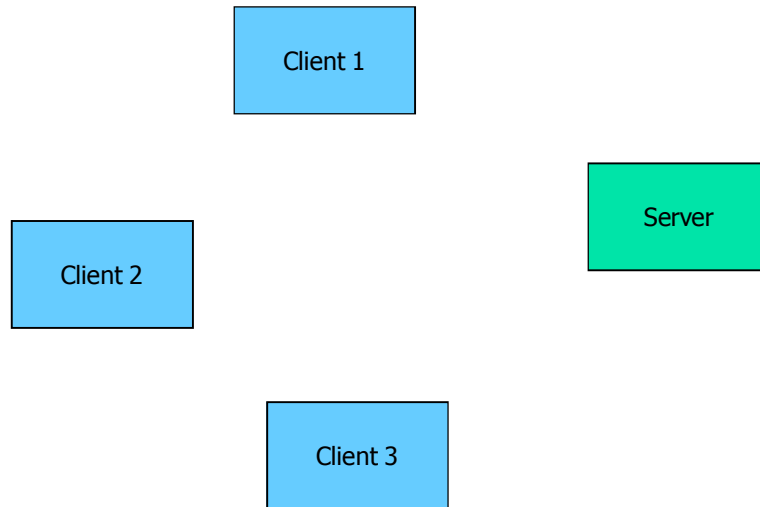
Introduction to Sockets

- n Used for interprocess communication
- n End-point for interprocess communication
 - n To send data
 - n To receive data
- n Client/Server model
 - n Two processes want to exchange data
 - n Server always waiting for client
 - n Client knows where the server is located
- n Connectionless vs. connection-oriented

How it circulates?

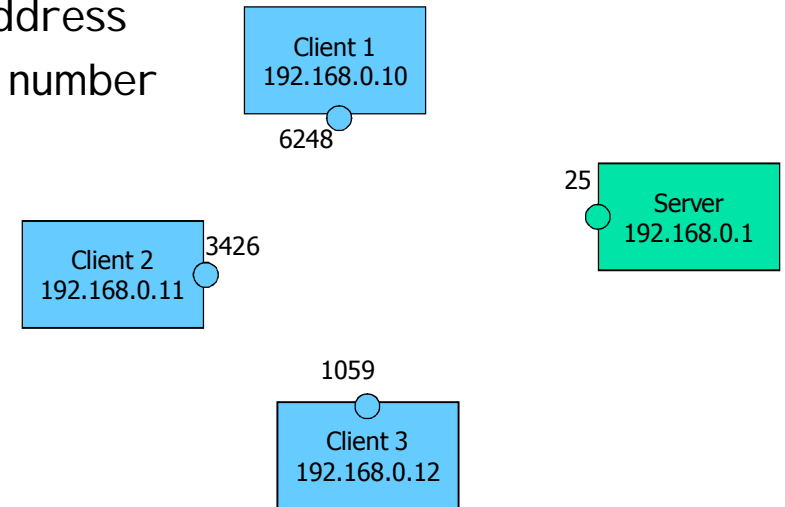


Use case



What makes a socket?

- n IP address
- n Port number



Types of addressess

- n Port numbers:
 - n Any integer from 0 to ...
 - n Range 0..1024 reserved
- n Types of addresses:
 - n Unicast
 - n Multicast
 - n Broadcast
 - n Loopback

Socket types

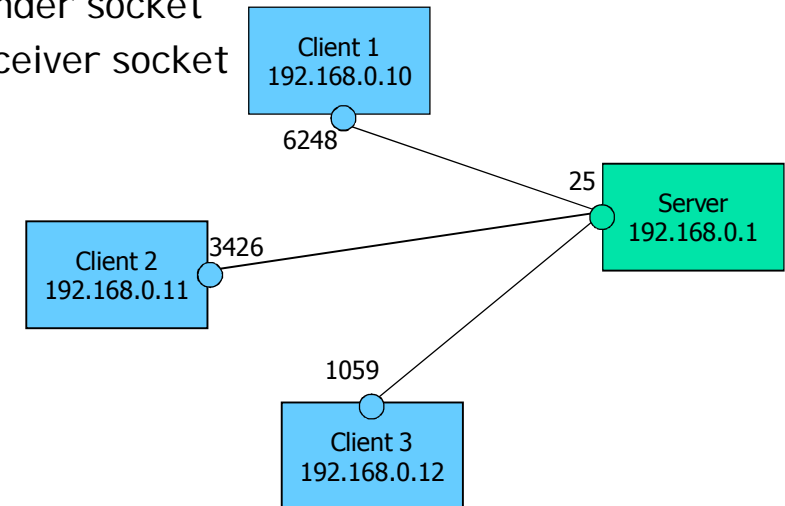
- n STREAM
 - n Based on TCP --> reliable, stream oriented
- n DATAGRAM
 - n Based on UDP -- > unreliable, message oriented
- n RAW
 - n Low-level socket
 - n Based "directly" on IP -- > no transport layer

Socket domains

- n AF_UNIX: address format is UNIX pathname
- n AF_INET: address format is host and port number

What makes a connection/communication?

- n Sender socket
- n Receiver socket



TCP Server

- n Get its IP and port number
 - n Usually provided in a configuration file or command line parameters
- n Create TCP Server Socket
- n Bind Server Socket to (IP, Port#)
- n Listen for new connections requests
- n Accept connections
 - n Create Client Socket
- n Exchange data
- n Close Client Socket/connection

TCP Client

- n Get Server IP and port number
 - n Usually provided as command line parameters
- n Create TCP Socket
- n Connect the socket to Server
 - n Server must be **up and listening** at that port, at that IP
- n Exchange data
- n Close socket/connection

UDP Server

- n Get its IP and port number
 - n Usually provided in a configuration file or command line parameters
- n Create UDP Server Socket
- n Bind Server Socket to (IP, Port#)
- n Exchange data
- n Close Client Socket/connection

UDP Client

- n Get Server IP and port number
 - n Usually provided as command line parameters
- n Create UDP Socket
- n Exchange data
- n Close socket/connection

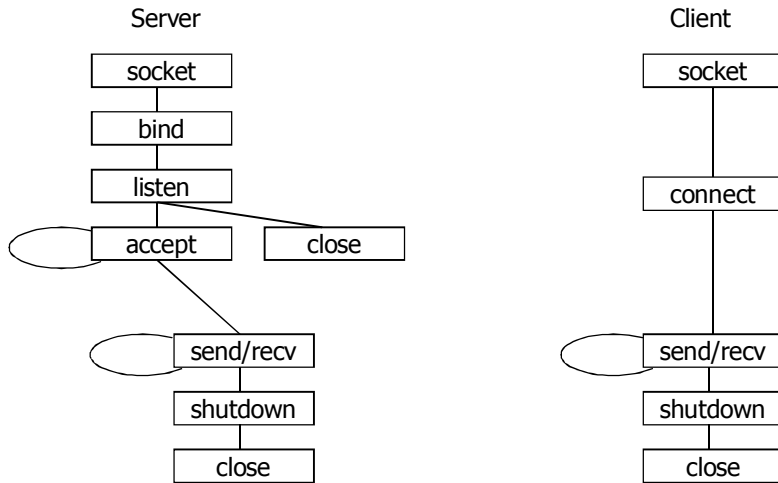
C/C++ Sockets API

- n `socket`: creates a socket
- n `bind`: assigns the socket
- n `listen`: wait for connections
- n `accept`: server accepts a connection request from a client
- n `connect`: client requests a connection request to a server
- n `send`, `sendto`: write to connection
- n `recv`, `recvfrom`: read from connection
- n `close`: end the communication

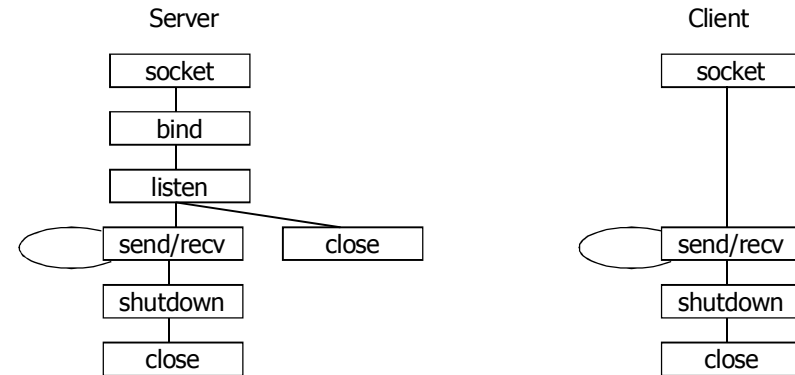
C/C++ Sockets API

- n `int socket(int domain, int type, int protocol)`
- n `int bind (int sid, struct sockaddr *addrPtr, int len)`
- n `int listen(int sid, int size)`
- n `int accept(int sid, struct sockaddr *addrPtr, int *lenPtr)`
- n `int connect(int sid, struct sockaddr *addrPtr, int len)`
- n `int send (int sid, const char *bufferPtr, int len, int flag)`
- n `int recv(int sid, char *bufferPtr, int len, int flags)`
- n `int shutdown(int sid, int how)`
- n `int close(int sid)`

TCP Server/Client



UDP Server/Client



TCP Server

```
sockaddr_in serverAddr;  
sockaddr_in &serverAddrCast = (sockaddr_in &) serverAddr ;  
// get a TCP/IP socket  
int listenFd = socket (AF_INET, SOCK_STREAM, 0);  
bzero(&serverAddr, sizeof (serverAddr ));  
serverAddr.sin_family = AF_INET;  
// any internet interface on this server .  
serverAddr.sin_addr.s_addr=htonl(INADDR_ANY);  
serverAddr.sin_port = htons (13);  
bind(listenFd, &serverAddrCast, sizeof (serverAddr ));  
listen (listenFd, 5);  
for ( ; ; ) {  
int connectFd = accept(listenFd,(sockaddr *) NULL, NULL);  
// .. read and write operations on connectFd ..  
shutdown(connectFd,2);  
close(connectFd);  
}
```

TCP Client

```
sockaddr_in serverAddr;  
sockaddr_in &serverAddrCast = (sockaddr_in &) serverAddr ;  
// get a tcp/ip socket  
int sockFd=socket(AF_INET,SOCK_STREAM,0);  
bzero(&serverAddr, sizeof ( serverAddr ));  
serverAddr . sin family = AF_INET;  
// host IP # in dotted decimal format !  
inet_pton(AF_INET,serverName,serverAddr.sin_addr);  
serverAddr.sin_port = htons (13);  
connect(sockFd,serverAddrCast, sizeof ( serverAddr ));  
// .. read and write operations on sockFd ..  
shutdown(sockFd,2);  
close(sockFd);
```

UDP Server

```
int socketId=socket(AF_INET, SOCK_DGRAM, 0);
sockaddr_in serverAddr,clientAddr;
sockaddr_in &serverAddrCast = (sockaddr_in &) serverAddr ;
sockaddr_in &clientAddrCast=(sockaddr_in &)clientAddr;
// allow connection to any addr on host
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(serverPort );
serverAddr.sin_addr.s_addr=INADDR_ANY;
// associate process with port
bind(socketId,&serverAddrCast, sizeof (addr ));
// receive from a client
int size = sizeof ( clientAddr );
recvfrom(socketId,buffer,bufferSize, 0, clientAddrCast,&size);
// reply to the client just received from
sendto(socketId,buffer,bufferSize, 0, clientAddrCast,size);
close(socketId);
```

UDP Client

```
int socketId=socket(AF_INET, SOCK_DGRAM, 0);
sockaddr_in serverAddr,clientAddr;
sockaddr_in &serverAddrCast = (sockaddr_in &) serverAddr ;
sockaddr_in &clientAddrCast=(sockaddr_in &)clientAddr;
// specify server address , port
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons( serverPort );
struct hostent *hp=gethostbyname(hostName);
memcpy(( char*)&serverAddr.sin_addr, (char*)hp ->h_addr,hp ->h_length);
// no need to bind if not peer to peer
int size = sizeof ( serverAddr );
sendto(socketId,buffer,bufferSize,0,
serverAddrCast,size);
recvfrom(socketId,buffer,bufferSize,0,
serverAddrCast,&size);
close(socketId);
```

Sockets and Java (TCP)

- n Open a socket
- n Open an input stream and output stream to the socket.
- n Read from and write to the stream
- n Close the streams.
- n Close the socket.

Open a Server Socket

```
ServerSocket serverSocket;
try {
serverSocket = new ServerSocket(PortNumber);
}
catch (IOException e) {
System.out.println(e);
}

Socket clientSocket = null;
try {
clientSocket = serverSocket.accept();
}
catch (IOException e) {
System.out.println(e);
}
```

Open a client socket

```
Socket echoClient;
try {
    echoClient = new Socket("Machine name", PortNumber);
}
catch (IOException e) {
    System.out.println(e);
}
```

Open a stream

n Server Side

```
PrintWriter out = new
PrintWriter(clientSocket.getOutputStream(), true);
BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
```

n Client side

```
PrintWriter out = new PrintWriter(echoSocket.getOutputStream(),
true);
BufferedReader in = new BufferedReader(new
InputStreamReader(echoSocket.getInputStream()));
```

Read/write from/to streams

n Read

- n in.readLine() // other methods are available, check class BufferedReader

n Write

- n Out.println(string s) //other methods are available, check class PrintWriter

Close streams/sockets

n Close streams first

- n in.close();
- n out.close()

n Then the sockets:

- n echoSocket.close()
- n clientSocket.close()
- n serverSocket.close()

TCP Java Server

```
import java.net.*;import java.io.*;
public class EchoServer {
    public static void main(String[] args) throws IOException {
        ServerSocket serverSocket = null;
        try {serverSocket = new ServerSocket(1234);}
        catch (IOException e) {system.err.println("Could not listen on : 1234.");
        System.exit(1);}
        Socket clientSocket = null;
        try {clientSocket = serverSocket.accept();}
        catch (IOException e) {System.err.println("Accept failed."); System.exit(1);}
        PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);   BufferedReader in = new
        BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
        String inputLine, outputLine;
        while ((inputLine = in.readLine()) != null) {outputLine = inputLine + " " + inputLine;
        out.println(outputLine);
        if (outputLine.equals("Bye. ")) break;}
        out.close(); in.close(); clientSocket.close();serverSocket.close();}
}
```

TCP Java Client

```
import java.io.*;import java.net.*;
public class EchoClient {
    public static void main(String[] args) throws IOException {   Socket echoSocket =
    null;PrintWriter out = null; BufferedReader in = null;
    String serverAddress = new String("127.0.0.1");
    try {echoSocket = new Socket(hostName, 1234);
    out = new PrintWriter(echoSocket.getOutputStream(), true);
    in = new BufferedReader(new InputStreamReader(
    echoSocket.getInputStream()));}
    catch (UnknownHostException e) {
    System.err.println("Don't know about host: " + serverAddress); System.exit(1);}
    catch (IOException e) {
    System.err.println("Couldn't get I/O for " + "the connection to: " +
    serverAddress);System.exit(1);}
    BufferedReader stdIn = new BufferedReader( new InputStreamReader(System.in));
    String userInput;while ((userInput = stdIn.readLine()) != null) {
    out.println(userInput);System.out.println("echo: " +
    in.readLine());}out.close();in.close();stdIn.close(); echoSocket.close(); }
}
```

Sending a datagram

```
public static void send(InetAddress dst, int port, byte[]
    outbuf, int len) {
    try {
        DatagramPacket request = new
            DatagramPacket(outbuf, len, dst, port);
        DatagramSocket socket = new DatagramSocket();
        socket.send(request);
    } catch (SocketException e) {
    } catch (IOException e) {
    }
}
```

Receiving a datagram

```
try {
    byte[] inbuf = new byte[256]; // default size
    DatagramSocket socket = new DatagramSocket();

    // Wait for packet
    DatagramPacket packet = new DatagramPacket(inbuf,
        inbuf.length);

    socket.receive(packet);

    // Data is now in inbuf
    int numBytesReceived = packet.getLength();
    } catch (SocketException e) {
    } catch (IOException e) {}
```



Quick demo

Conclusion

- n What is a socket?
- n Connection-oriented/connectionless and sockets
- n API in java and C/C++
- n Examples

References

- n <http://beej.us/guide/bgnet/>
- n <http://java.sun.com/docs/books/tutorial/networking/> (the best you can start with)
- n Plenty of handouts on the net
 - n Just google it !