NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCE
Computer Networks Lab (CL307)
Lab Session 02

TCP/IP and the OSI Model

The networking industry has a standard seven-layer model for network protocol architecture called the Open Systems Interconnection (OSI) model. The OSI model represents an effort by the International Organization for Standardization (ISO), an international standards organization, to standardize the design of network protocol systems to promote interconnectivity and open access to protocol standards for software developers.

TCP/IP was already on the path of development when the OSI standard architecture appeared and, strictly speaking, TCP/IP does not conform to the OSI model. However, the two models did have similar goals, and enough interaction occurred among the designers of these standards that they emerged with a certain compatibility. The OSI model has been very influential in the growth and development of protocol implementations, and it is quite common to see the OSI terminology applied to TCP/IP.

OSI MODEL	TCP/IP MODEL	PROTOCOLS
Application Layer	Application Layer	TELNET,SSH,DHCP,SMTP,DNS,FTP,TFTP,POP3,HTTP,HTTPS
Presentation Layer		SSL
Session Layer		
Transport layer	Transport layer	TCP,UDP
Network Layer	Network Layer	IPV4,IPV6,ICMP,IGMP
Datalink Layer	Datalink Layer	ARP,RARP,PPP
Physical Layer	Physical Layer	Ethernet

INTERNET PROTOCOL (IP)

An IP address (abbreviation of Internet Protocol address) is an identifier assigned to each computer and other device (e.g., printer, router, mobile device, etc.) connected to a TCP/IP network that is used to locate and identify the node in communications with other nodes on the network. IP addresses are usually written and displayed in human-readable notations, such as 172.16.254.1 in IPv4, and 2001:db8:0:1234:0:567:8:1 in IPv6.

Version 4 of the Internet Protocol (IPv4) defines an IP address as a 32-bit number. However, because of the growth of the Internet and the depletion of available IPv4 addresses, a new version of IP (IPv6), using 128 bits for the IP address, was developed.

IP Address Classes

Class A	1 – 127	(Network 127 is reserved for loopback and internal testing) Leading bit pattern	0	00000000.00000000.00000000.00000000 Network . Host . Host . Host
Class B	128 – 191	Leading bit pattern	10	10000000.00000000.00000000.00000000 Network . Network . Host . Host
Class C	192 – 223	Leading bit pattern	110	11000000.00000000.00000000.00000000 Network . Network . Network . Host
Class D	224 – 239	(Reserved for multicast)		
Class E	240 – 255	(Reserved for experimental, used for research)		

Private Address Space

Class A	10.0.0.0 to 10.255.255.255
Class B	172.16.0.0 to 172.31.255.255
Class C	192.168.0.0 to 192.168.255.255

Default Subnet Masks

Class A	255.0.0.0
Class B	255.255.0.0
Class C	255.255.255.0

IP address Classes

Class	# Network Bits	# Hosts Bits	Decimal Address Range	Subnet mask
Class A	8 bits	24 bits	1-126	255.0.0.0
Class B	16 bits	16 bits	128-191	255.255.0.0
Class C	24 bits	8 bits	192-223	255.255.255.0
Class D	Reserved for Multicasting		224-239	N/A
Class E	Reserved for R & D		240-255	N/A

Address Class Identification

Address	Class
10.250.1.1	A
150.10.15.0	B
192.14.2.0	_____
148.17.9.1	_____
193.42.1.1	_____
126.8.156.0	_____
220.200.23.1	_____
230.230.45.58	_____
177.100.18.4	_____
119.18.45.0	_____
249.240.80.78	_____
199.155.77.56	_____
117.89.56.45	_____
215.45.45.0	_____
199.200.15.0	_____
95.0.21.90	_____
33.0.0.0	_____
158.98.80.0	_____
219.21.56.0	_____

Network & Host Identification

Circle the network portion of these addresses:

177.100.18.4

119.18.45.0

209.240.80.78

199.155.77.56

117.89.56.45

215.45.45.0

192.200.15.0

95.0.21.90

33.0.0.0

158.98.80.0

217.21.56.0

10.250.1.1

150.10.15.0

192.14.2.0

148.17.9.1

193.42.1.1

126.8.156.0

220.200.23.1

Circle the host portion of these addresses:

10.15.123.50

171.2.199.31

198.125.87.177

223.250.200.222

17.45.222.45

126.201.54.231

191.41.35.112

155.25.169.227

192.15.155.2

123.102.45.254

148.17.9.155

100.25.1.1

195.0.21.98

25.250.135.46

171.102.77.77

55.250.5.5

218.155.230.14

10.250.1.1

Network Addresses

Using the IP address and subnet mask shown write out the network address:

188.10.18.2
255.255.0.0 — 188 . 10 . 0 . 0

10.10.48.80
255.255.255.0 — 10 . 10 . 48 . 0

192.149.24.191
255.255.255.0 — _____

150.203.23.19
255.255.0.0 — _____

10.10.10.10
255.0.0.0 — _____

186.13.23.110
255.255.255.0 — _____

223.69.230.250
255.255.0.0 — _____

200.120.135.15
255.255.255.0 — _____

Host Addresses

Using the IP address and subnet mask shown write out the host address:

188.10.18.2
255.255.0.0 — 0.0.18.2

10.10.48.80
255.255.255.0 — 0.0.0.80

222.49.49.11
255.255.255.0 _____

128.23.230.19
255.255.0.0 _____

10.10.10.10
255.0.0.0 _____

200.113.123.11
255.255.255.0 _____

223.169.23.20
255.255.0.0 _____

203.20.35.215
255.255.255.0 _____

Default Subnet Masks

Write the correct default subnet mask for each of the following addresses:

177.100.18.4	255 . 255 . 0 . 0
119.18.45.0	255 . 0 . 0 . 0
191.249.234.191	
223.23.223.109	
10.10.250.1	
126.123.23.1	
223.69.230.250	
192.12.35.105	
77.251.200.51	
189.210.50.1	

Introduction to Socket Programming

The java.net package of the J2SE APIs contains a collection of classes and interfaces that provide the low-level communication details, allowing you to write programs that focus on solving the problem at hand.
The java.net package provides support for the two common network protocols:

- **TCP: TCP stands for Transmission Control Protocol, which allows for reliable communication between two applications. 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:

Sockets provide the communication mechanism between two computers using TCP. A client program creates a socket on its end of the communication and attempts to connect that socket to a server.
When the connection is made, the server creates a socket object on its end of the communication. The client and server can now communicate by writing to and reading from the socket.

- **The java.net.Socket class represents a socket**
- **The java.net.ServerSocket class provides a mechanism for the server program to listen for clients and establish connections with them.**

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.**
3. **After the server is waiting, a client instantiates a Socket object, specifying the server name and port number to connect to.**
4. **The constructor of the Socket class attempts to connect the client to the specified server and 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.**

After the connections are established, communication can occur using I/O streams. Each socket has both an OutputStream and an InputStream.
The client's OutputStream is connected to the server's InputStream, and the client's InputStream is connected to the server's OutputStream.
TCP is a two way communication protocol, so data can be sent across both streams at the same time. There are following useful classes providing complete set of methods to implement sockets.

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:

SN	Methods with Description
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

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.

Here are some of the common methods of the ServerSocket class:

SN	Methods with Description
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
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 instantiated the ServerSocket using the no-argument constructor.

When the ServerSocket invokes accept(), the method does not return until a client connects. After a client does connect, the ServerSocket creates a new Socket on an unspecified port and returns a reference to this new Socket. A TCP connection now exists between the client and server, and communication can begin.

Socket Class Methods:

The java.net.Socket class represents the socket that both the client and 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.
The Socket class has five constructors that a client uses to connect to a server:

SN	Methods with Description
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.
3	public Socket(String host, int port, InetAddress localAddress, int localPort) throws IOException. Connects to the specified host and port, creating a socket on the local host at the specified address and port.
4	public Socket(InetAddress host, int port, InetAddress localAddress, int localPort) throws IOException. This method is identical to the previous constructor, except that the host is denoted by an InetAddress object instead of a String
5	public Socket() Creates an unconnected socket. Use the connect() method to connect this socket to a server.

When the Socket constructor returns, it does not simply instantiate a Socket object but it actually attempts to connect to the specified server and port.

Some methods of interest in the Socket class are listed here. Notice that both the client and server have a Socket object, so these methods can be invoked by both the client and server.

SN	Methods with Description
1	public void connect(SocketAddress host, int timeout) throws IOException This method connects the socket to the specified host. This method is needed only when you instantiated the Socket using the no-argument constructor.
2	public InetAddress getInetAddress() This method returns the address of the other computer that this socket is connected to.
3	public int getPort() Returns the port the socket is bound to on the remote machine.
4	public int getLocalPort() Returns the port the socket is bound to on the local machine.
5	public SocketAddress getRemoteSocketAddress() Returns the address of the remote socket.
6	public InputStream getInputStream() throws IOException Returns the input stream of the socket. The input stream is connected to the output stream of the remote socket.
7	public OutputStream getOutputStream() throws IOException Returns the output stream of the socket. The output stream is connected to the input stream of the remote socket
8	public void close() throws IOException Closes the socket, which makes this Socket object no longer capable of connecting again to any server

InetAddress Class Methods:

This class represents an Internet Protocol (IP) address. Here are following useful methods which you would need while doing socket programming:

SN	Methods with Description
1	static InetAddress getByAddress(byte[] addr) Returns an InetAddress object given the raw IP address .
2	static InetAddress getByAddress(String host, byte[] addr) Create an InetAddress based on the provided host name and IP address.
3	static InetAddress getByName(String host) Determines the IP address of a host, given the host's name.
4	String getHostAddress() Returns the IP address string in textual presentation.
5	String getHostName() Gets the host name for this IP address.
6	static InetAddress InetAddress getLocalHost() Returns the local host.
7	String toString() Converts this IP address to a String.