Lecture (04)
Network Programming (I)

Dr. Ahmed ElShafee

Agenda

- Concepts of socket programming
- Network Layers
- TCP and UDP
- Ports
- Sockets
- Email
- Internet Addressing
- Retrieving Data from a URL

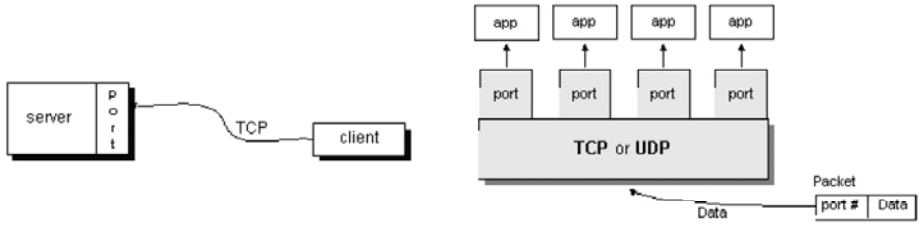
Concepts of socket programming

Important concepts needed for writing network program in Java

1) Communication protocols: TCP and UDP
2) Ports and Internet Addresses
3) Sockets
4) Uniform Resource Locator (URL)
5) Uniform Resource Identifier (URI)
6) Streams and Threads
7) Classes in java.net and java.io packages

Ports

- Ports are a virtual channels that enables applications to share the single network channel to exchange data.
- Port numbers are represented by 16-bit numbers. (0 to 65,535) The port numbers ranging from 0 - 1023 reserved for use by well known application



Sockets

You can reach required service via its network and port IDs. what then?

a) If you are a client

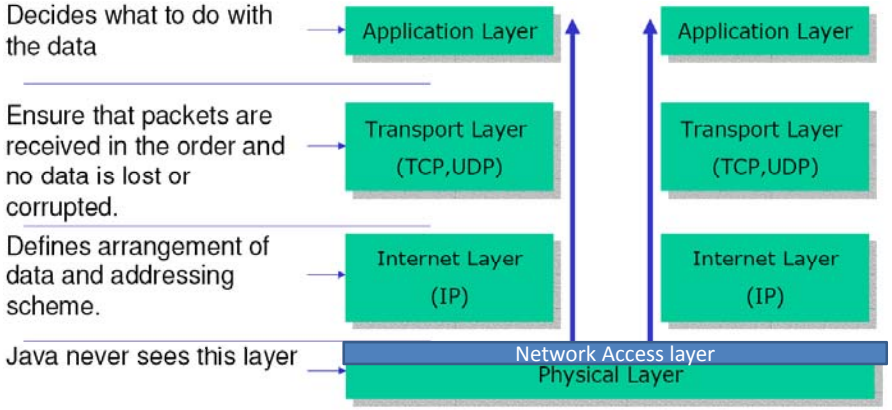
- you need an API that will allow you to send messages to that service and read replies from it

b) If you are a server

- you need to be able to create a port and listen at it.
- you need to be able to read the message comes in and reply to it.

The **Socket** and **ServerSocket** are the Java client and server classes to do this.

Network Layers



TCP and UDP

- Java only supports TCP (Transmission Control Protocol), UDP (User Datagram Protocol) and application layer protocols built on top of these.

Simple client example

```
import java.io.*;
import java.net.*;
public class tcpclient
    {
    public static void main(String[] args) throws IOException
            {
            Socket clientSocket = null;
            clientSocket = new Socket("192.168.1.2",5050);
            Writer w = new OutputStreamWriter(clientocket.getOutputStream(), true);
            InputStreamReader isr = new InputStreamReader
    (clientSocket.getInputStream());
             BufferedReader br = new BufferedReader (isr);
             System.out.print ("Connection established.\n>");
             w.write("Connection Established.\r\n>");
                    }}
```

Email protocols

IMAP (Internet Message Access Protocol) –

Is a standard protocol for accessing e-mail from your local server.

IMAP is a client/server protocol in which e-mail is received and held for you by your Internet server.

As this requires only a small data transfer this works well even over a slow connection such as a modem.

Only if you request to read a specific email message will it be downloaded from the server.

You can also create and manipulate folders or mailboxes on the server, delete messages etc.

Email protocols (2)

The **POP (Post Office Protocol 3)** protocol provides a simple, standardized way for users to access mailboxes and download messages to their computers.

When using the POP protocol all your eMail messages will be downloaded from the mail server to your local computer.

You can choose to leave copies of your eMails on the server as well.

Email protocols (3)

The **SMTP (Simple Mail Transfer Protocol)** protocol is used by the Mail Transfer Agent (MTA) to deliver your eMail to the recipient's mail server.

The SMTP protocol can only be used to send emails, not to receive them.

Depending on your network / ISP settings, you may only be able to use the SMTP protocol under certain conditions

Email protocols (4)

The **HTTP protocol** is not a protocol dedicated for email communications, but it can be used for accessing your mailbox.

Also called web based email, this protocol can be used to compose or retrieve emails from an your account.

Hotmail is a good example of using HTTP as an email protocol.

Email

- SMTP E-mail is sent by socket communication with port 25 on a computer system.
- Open a socket connected to port 25 on some system, and speak “mail protocol” to the daemon at the other end.

Email (2)

Using telnet

telnet 192.168.1.2 25

<<220 192.168.1.2 ESMTP (Code-Crafters Ability Mail Server 2.52)

HELO 192.168.1.2

<<250 192.168.1.2

MAIL FROM: user001@192.168.1.2

<<250 Email address accepted. <user001@192.168.1.2>

RCPT TO: user002@192.168.1.2

<< 250 Email address accepted. <user002@192.168.1.2>

Email (3)

DATA

<<354 Please send the data and end with a <CRLF>.<CRLF>.

SUBJECT: email subject<ret><ret>

Body of the test email

.

<<250 Mail accepted and queued for delivery

quit

Sendemail.java

```
import java.io.*;
import java.net.*;
public class sendemail
    {
    public static void main(String args[]) throws    IOException
            {
            Socket sock;
            DataInputStream dis;
            BufferedReader br;
            PrintStream ps;
            System.out.println(">>> Connect 192.168.1.2");
            sock = new Socket("192.168.1.2", 25);
            br = new BufferedReader (new InputStreamReader(sock.getInputStream()));
```

Sendemail.java (2)

```
ps = new PrintStream( sock.getOutputStream());
System.out.println( br.readLine() );
System.out.println(">>> HELO 192.168.1.2");
ps.println("HELO 192.168.1.2");
System.out.println( br.readLine() );
System.out.println(">>> MAIL FROM: user001@192.168.1.2");
ps.println("MAIL FROM: <user001@192.168.1.2>");
System.out.println( br.readLine() );
String Addressee= "user002@192.168.1.2";
System.out.println(">>> Rcpt to: " + Addressee);
ps.println("RCPT TO: <" + Addressee +">");
System.out.println( br.readLine() );
System.out.println(">>> Send \"data\"");
ps.println("DATA");
```


Sendemail.java (3)

```
System.out.println( br.readLine() );
System.out.println(">>>>>>>>>>");
System.out.println(">>> Subject: test email\n\n");
System.out.println(">>> This is the message\n thatJava sent");
System.out.println(">>> We are testing Socket Programming");
System.out.println(">>> in ACU-CSIT.");
System.out.println(">>>>>>>>>>");
ps.println("SUBJECT:Test email\n");
ps.println("This is the message\n that Java sent");
ps.println("We are testing Socket Programming");
ps.println("in ACU-CSIT Training program.");
System.out.println(">>> .");
ps.println(".");
System.out.println( br.readLine() );
```

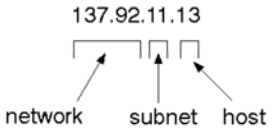

Sendemail.java (4)

```
ps.println("QUIT");
System.out.println( br.readLine() );
ps.flush();
sock.close();
    }
}
```


Internet Addressing

Internet address (IP address) is a unique number for identifying a device connected to the Internet.

The current standard is IPv4 which are four bytes long.



The hostname and IP address is, in Java, represented by java.net.InetAddress.

InetAddress is used by many other networking classes, including Socket, ServerSocket, URL, DatagramSocket, DatagramPacket, and more.

Internet Addressing (2)

InetAddress

methods:

a) static InetAddress getByName(String host)

b) static InetAddress getLocalHost() //local pc name/ip

c) String getHostAddress(); // return host ip

d) String getHostName(); return host name

urlbyname.java

```
InetAddress address = InetAddress.getByName (args[0]) ;
System.out.println(address);
```

```
java urlbyname www.draelshafee.net
www.draelshafee.net/97.74.144.100
```

```
InetAddress Address = InetAddress.getLocalHost();
System.out.println(Address);
```

```
java localhost
Acu-skpv5wv1xst/192.168.1.6
```


Internet Addressing (3)

The URL Class

The java.net.URL is an abstraction of a Uniform Resource Locator (URL).

URLs are composed of five pieces:

1. The scheme, also known as the protocol
2. The authority
3. The path
4. The query string
5. The fragment identifier, also known as the section or ref

<scheme>://<authority><path>?<query>#<fragment>

Internet Addressing (4)

For example, given the URL :

http://www.ibiblio.org/javafaq/javabooks/index.html?isbn=123456789#toc

1. scheme : http
2. authority : www.ibiblio.org
3. path : /javafaq/books/javabooks/index.html
4. query string : isbn=123456789
5. fragment identifier : toc

Internet Addressing (5)

The authority may further be divided into the user info, the host, and the port.

For example, in the URL
http://admin@www.blackstar.com:8080/

1. user info : admin
2. host : www.blackstar.com
3. port : 8080

Internet Addressing (6)

- The java.net.URL class provides static methods for getting the above
- mentioned information:
- a) getFile( )
- b) getHost()
- c) getPort( )
- d) getProtocol()
- e) getRef()
- f) getQuery( )
- g) getPath( )
- h) getUserInfo( )
- i) getAuthority( )

Internet Addressing (7)

you can construct instances of java.net.URL using one of its constructors.

1) public URL(String url) throws MalformedURLException
2) public URL(String protocol, String hostname, String file) throws MalformedURLException
3) public URL(String protocol, String host, int port, String file) throws MalformedURLException

ProtocolTesterdemo.java (1)

```
try
        {
        URL u = new URL(url);
        System.out.println(u.getProtocol( ) + " is supported");
        }
catch (MalformedURLException ex)
        {
        String protocol = url.substring(0, url.indexOf(':'));
        System.out.println(protocol + " is not supported");
```

Retrieving Data from a URL

The URL class provides methods for retrieving data from a URL:

public InputStream openStream( ) throws IOException

public URLConnection openConnection( ) throws IOException

public Object getContent( ) throws IOException

Retrieving Data from a URL (2)

Procedure to use the methods:

1) Create an URL object

e.g. URL u = new URL("http://www.iist.unu.edu");

2) Open an InputStream object directly from the URL object

e.g. InputStream in = u.openStream( );

3) Or open an URLConnection object from the URL object and then get an InputStream object from the URLConnection object .

e.g. URLConnection uc = u.openConnection( );

InputStream in = uc.getInputStream( );

Retrieving Data from a URL (3)

4) In either case, you will have an InputStream. What's followed is the normal I/O procedure for getting data.

5) Don't forget to put the try catch block for catching the *MalformedURLException* and *IOException*.

Retrieving Data from a URL (4)

What is the difference between using the **openStream** and **openConnection** method?

1) **openStream** method only give you the access to the raw data and cannot detect the encoding information.

2) **openConnection** method opens a socket to the specified URL and returns a URLConnection object.

3) The **URLConnection** object gives you access to everything sent by the server. You can access all the metadata specified by the protocol such as the scheme. The URLConnection class also lets you write data to as well as read from a URL.

Retrieving Data from a URL (5)

The following methods are used to access the header fields and the contents after the connection is made to the remote object:

1)getContent

2)getHeaderField

3)getInputStream

4)getOutputStream

Retrieving Data from a URL (6)

Certain header fields are accessed frequently. The methods:

1) getContentEncoding

2) getContentLength

3) getContentType

4) getDate

5) getExpiration

6) getLastModifed

Urlconnectionreader1.java

```
URL site = new URL(args[0]);
URLConnection sc = site.openConnection();
BufferedReader in = new BufferedReader(new InputStreamReader
(sc.getInputStream()));
String inputLine;
while ((inputLine = in.readLine()) != null)
System.out.println(inputLine);
in.close();}}
```

```
java urlconnectionreader1 http://www.draelshafee.net
```


output

```
java urlconnectionreader1 http://www.draelshafee.net
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1"/>
<meta name="description" content="description"/>
<meta name="keywords" content="keywords"/>
<meta name="author" content="author"/>
<link rel="stylesheet" type="text/css" href="default.css" media="screen"/>
<title>Dr. Ahmed ElShafee</title>
</head>
<body>
```

Urlconnectionreader2.java

```
        URL site = new URL(args[0]);
        URLConnection sc = site.openConnection();
        BufferedReader in = new BufferedReader(new InputStreamReader
   (sc.getInputStream()));
        String inputLine;
BufferedWriter out = new BufferedWriter(new FileWriter(args[1]));
        while ((inputLine = in.readLine()) != null)
                out.write(inputLine);
```


output

```
java urlconnectionreader2 http://www.draelshafee.net c:\data\index.html
```

C:\data\index.html

Dr. Ahmed ElShafee,
Home Fall2010 Spring2011 ACUCSIT portal
welcome,..

Ahram Canadian Uiversity, Faculty of CSIT portal
Fall 2010 results
Cumulative GPA till Fall 2010
Contacts
Email: mail@draelshafee.net
Web: www.draelshafee.net

© 2011Dr. Ahmed ElShafee web site. Design by Dr. Ahmed ElShafee

Thanks,
See you next Week, isA