Network Programming [CACS355]

BCA 6th Sem

Er. Sital Prasad Mandal

(Email : info.sitalmandal@gmail.com)
Bhadrapur, Jhapa, Nepal

https://networkprogam-mmc.blogspot.com/

Unit-6

Sockets for Clients

Unit-6

Sockets for Clients

https://wq.apnic.net/static/search.html
https://www.site24x7.com/find-ip-address-of-web-site.html
https://who.is/dns/fb.com

```
>cmd
>telnet whois.apnic.net 43
                                157.240.19.35  - fb.com : Ip
>telnet time.nist.gov  13
                132.163.96.6 - time.nist.gov :IP
```

Unit-6

Sockets for Clients

1. A **socket** is a reliable connection for the transmission of data between two hosts.
2. Sockets isolate programmers from the details of packet encodings, lost and retransmitted packets, and packets that arrive out of order.
3. There are limits. Sockets are more likely to throw **IOExceptions** than files.

There are four fundamental **operations a socket performs:**

- Connect to a remote machine
- Send data
- Receive data
- Close the connection

Unit-6

Sockets for Clients

The java.net.Socket class

- The java.net.Socket class allows you to create socket objects that perform all four fundamental socket operations.
- You can connect to remote machines; you can send data; you can receive data; you can close the connection.
- Each Socket object is associated with exactly one remote host. To connect to a different host, you must create a new Socket object.

Unit-6

Sockets for Clients

Constructing a Socket

Connection is accomplished through the constructors:

1. public Socket(String host, int port) throws UnknownHostException, IOException
2. public Socket(InetAddress address, int port) throws IOException
3. public Socket(String host, int port, InetAddress localAddr, int localPort) throws IOException
4. public Socket(InetAddress address, int port, InetAddress localAddr, int localPort) throws IOException

- The Socket() constructors do not just create a Socket object. They also attempt to connect the underlying socket to the remote server.

Unit-6

Reading Input from a Socket

The following code fragment connects to the daytime server on port 13 of time.nist.gov, and displays the data it sends. TimeClient

```
public class TimeClient {
    public static void main(String[] args) {
        try {
            Socket s = new Socket("time.nist.gov" , 13);
            InputStream in = s.getInputStream();
            InputStreamReader isr = new InputStreamReader(in, "ASCII");
            BufferedReader br = new BufferedReader(isr);
             br.lines().forEach(System.out::println);
             //System.out.println(new Date().toString());
//            int line;
//            while ((line = br.read()) != -1) {
//                System.out.print((char) line);
//            }
            System.out.println();
        }
        catch (IOException e) {
            System.out.println(e);
        }
    }
}
```

```
59762 22-07-02 09:42:26 50 0 0 325.8 UTC(NIST) *
```

Unit-6

Reading Input from a Socket

PortScanner

```
public class LocalPortScanner {
    public static void main(String[] args) throws UnknownHostException {
       InetAddress remote = InetAddress.getLocalHost();
        String hostname = remote.getHostName();
         for (int port = 1; port < 65536; port++) {
           try {
                Socket s = new Socket(remote, port);
                System.out.println("A server is listening on port "+ port + " of " + hostname);
                s.close();
                }
            catch (IOException ex) {
                    // The remote host is not listening on this port
                }
            }
    }
}
```

Unit-6

8.3 Getting Information About a Socket

Getter methods of Socket class

```
public InetAddress getInetAddress()
public int getPort()
public InetAddress getLocalAddress()
public int getLocalPort()
```

Address and port of the remote host

8.3 Getting Information About a Socket

Get a socket's information

```
public class SocketInfo {
    public static void main(String[] args) {
        String urlhost = "spm.com.np";  // take only Url/domain
        try {
            Socket theSocket = new Socket(urlhost, 80);
            System.out.println("Connected to " + theSocket.getInetAddress()
                    + " on port " + theSocket.getPort() + " from port "
                    + theSocket.getLocalPort() + " of "
                    + theSocket.getLocalAddress());
        } catch (UnknownHostException ex) {
            System.err.println("I can't find " + urlhost);
        } catch (SocketException ex) {
            System.err.println("Could not connect to " + urlhost);
        } catch (IOException ex) {
            System.err.println(ex);
        }
    }
}
```

https://networkprogam-mmc.blogspot.com/

Unit-6

Socket Options

- Nine options for client-side sockets
  - Follow classic Unix C naming conventions
  - TCP_NODELAY, SO_BINDADDR, SO_TIMEOUT, SO_LINGER SO_SNDBUF, SO_RCVBUF, SO_KEEPALIVE, OOBINLINE, IP_TOS
- TCP_NODELAY
  - Packets are sent as quickly as possible regardless of their size (even 1 byte)

```
public void setTcpNoDelay(boolean on) throws SocketException
public boolean getTcpNoDelay() throws SocketException
```

  - setTcpNoDelay(true) turns off buffering for the socket

```
if (!s.getTcpNoDelay()) s.setTcpNoDelay(true);
```

- SO_LINGER
  - Specifies what to do with unsent data when a socket is closed

```
public void setSoLinger(boolean on, int seconds) throws SocketException
public int getSoLinger() throws SocketException
```

  - By default, close() returns immediately and the system still tries to send any remaining data
  - If the linger time is set to 0, any unsent packets are thrown away when closed
  - If the linger time > 0 and SO_LINGER is turned on, close() blocks while waiting the specified number of seconds for the data to be sent
    - Maximum linger time is 65,535 seconds
    - getTcpSoLinger() returns -1 if this option is disabled

```
if (s.getTcpSoLinger() == -1) s.setSoLinger(true, 240);
```

Unit-6

Socket Options

- SO_KEEPALIVE
  - If turned on, the client occasionally sends a data packet over an idle connection to make sure the server hasn't crashed; The default is false

    ```
    public void setKeepAlive(boolean on) throws SocketException
    public boolean getKeepAlive() throws SocketException
    ```

    - Most commonly once every two hours
      - If the server fails to respond, the client keeps trying for a little more than 11 minutes
        - »If not response within 12 minutes, the client closes the socket

    ```
    if (s.getKeepAlive()) s.setKeepAlive(false);
    ```
- OOBINLINE
  - Sends a single byte of '*urgent*' data out of band; The default is off

    ```
    public void setOOBInline(boolean on) throws SocketException
    public boolean getOOBInline() throws SocketException

    public void sendUrgentData(int data) throws IOException
    ```
  - More modern approach is to place the urgent data in the regular received data queue in its proper order
    - Java does not distinguish it from non-urgent data

    ```
    if (!s.getOOBInline()) s.setOOBInline(true);
    ```
- SO_REUSEADDR
  - If turned on, another socket is allowed to bind to the port even while data may be outstanding for the previous socket
    - setReuseAddress() must be called before the new socket binds to the port
    - The default is off; It may not immediately release the local port when the socket was closed

    ```
    public void setReuseAddress(boolean on) throws SocketException
    public boolean getReuseAddress() throws SocketException
    ```

https://networkprogam-mmc.blogspot.com/

Unit-6

Socket Options

- IP_TOS Class of Service
  - Specifies the class of service; stored in an eight-bit field in the IP header (0-255)

```
public int getTrafficClass() throws SocketException
public void setTrafficClass(int trafficClass) throws SocketException
```

    - High-order six bits contain a Differentiated Services Code Point (DSCP)
      - Up to 2^6 different traffic classes; Not hard and fast guarantees of services
      - Respected on some networks internally; A packet crosses ISPs is almost always ignored
    - Low-order two bits contain an Explicit Congestion Notification (ECN) value
      - Should be set to zero; usually set by intermediate routers
  - Many implementations ignore these values completely
    - Android treats it as a no-op

```
Socket s = new Socket("www.yahoo.com", 80);
s.setTrafficClass(0xB8); // 10111000 in binary
```

Table 8-1. Common DSCP values and interpretations

PHB (Per Hop Behavior)	Binary value	Purpose
Default	00000	Best-effort traffic.
Expedited Forwarding (EF)	101110	Low-loss, low-delay, low-jitter traffic. Often limited to 30% or less of network capacity.
Assured Forwarding (AF)	*multiple*	Assured delivery up to a specified rate.
Class Selector	xxx000	Backward compatibility with the IPv4 TOS header, as stored in the first three bits.

```
Socket s1 = new Socket("www.example.com", 80);
s1.setTrafficClass(0x26); // 00100110 in binary
Socket s2 = new Socket("www.example.com", 80);
s2.setTrafficClass(0x0A); // 00001010 in binary
Socket s3 = new Socket("www.example.com", 80);
s3.setTrafficClass(0x0E); // 00001110 in binary
```

Table 8-2. Assured forwarding priority classes

	Class 1 (lowest priority)	Class 2	Class 3	Class 4 (highest priority)
Low Drop Rate	AF11 (001010)	AF21 (010010)	AF31 (011010)	AF41 (100010)
Medium Drop Rate	AF12 (001100)	AF22 (010100)	AF32 (011100)	AF42 (100100)
High Drop Rate	AF13 (001110)	AF23 (010110)	AF33 (011110)	AF43 (100110)

- Use setPerformancePreferences() instead to assign preferences

```
public void setPerformancePreferences(int connectionTime, int latency, int bandwidth)
```

    - i.e. setPerformancePreferences(2, 1, 3) means Maximum bandwidth is most important

Unit-6

Socket Options

Several methods set various socket options. Most of the time the defaults are fine.

```
public void setTcpNoDelay(boolean on) throws SocketException
public boolean getTcpNoDelay() throws SocketException
public void setSoLinger(boolean on, int val) throws SocketException
public int getSoLinger() throws SocketException
public void setSoTimeout(int timeout) throws SocketException
public int getSoTimeout() throws SocketException
```

Unit-6

Socket Exceptions

- Most methods of the Socket class are declared to throw IOException or its subclass, java.net.SocketException

```
public class SocketException extends IOException
```

  – Several subclasses of SocketException that provide more information about what went wrong and why

```
public class BindException extends SocketException
public class ConnectException extends SocketException
public class NoRouteToHostException extends SocketException

public class ProtocolException extends IOException
```

  - BindException: a local port is in use or no privileges to use
  - ConnectException: connection refused at the remote
  - NoRouteToHostException: connection timed out
  - ProtocolException: received data violates the TCP/IP specification

Whois Prefixes

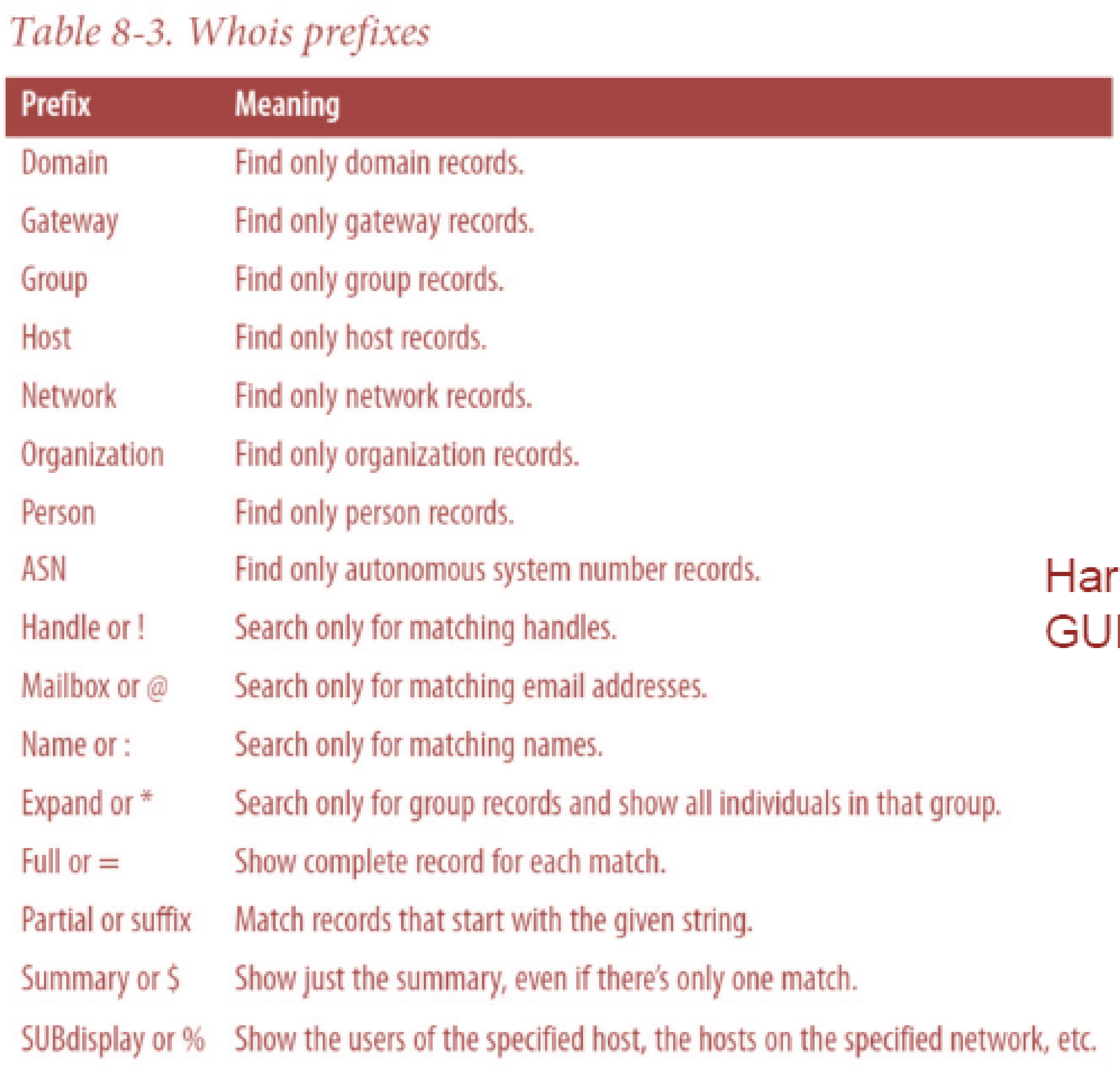

Table 8-3. Whois prefixes

Prefix	Meaning
Domain	Find only domain records.
Gateway	Find only gateway records.
Group	Find only group records.
Host	Find only host records.
Network	Find only network records.
Organization	Find only organization records.
Person	Find only person records.
ASN	Find only autonomous system number records.
Handle or !	Search only for matching handles.
Mailbox or @	Search only for matching email addresses.
Name or :	Search only for matching names.
Expand or *	Search only for group records and show all individuals in that group.
Full or =	Show complete record for each match.
Partial or suffix	Match records that start with the given string.
Summary or $	Show just the summary, even if there's only one match.
SUBdisplay or %	Show the users of the specified host, the hosts on the specified network, etc.

Hard to remember the prefixes;
GUI to help users specify them

https://networkprogam-mmc.blogspot.com/

Unit-6

Sockets for Clients

8.1 Using Sockets

– Socket, SimpleDateFormat

– Daytime (Examples 8-1 and 8-2), Time (Example 8-3), DictClient (Example 8-4)

8.2 Constructing and Connecting Sockets

– LowPortScanner (Example 8-5)

8.3 Getting Information about Sockets

– SocketInfo (Example 8-6)

8.4 Setting Socket Options

– TCP_NODELAY, SO_BINDADDR, SO_TIMEOUT, SO_LINGER
SO_SNDBUF, SO_RCVBUF, SO_KEEPALIVE, OOBINLINE, IP_TOS

8.5 Socket Exceptions