Java Network Programming UDP Support

Dave Price

Computer Science

University of Wales, Aberystwyth

UDP Socket Programming

- Classes etc. contained within java.net
- UDP datagrams are sent in a connectionless manner, each fully addresses and sent quite independently of each other
- UDP support from Java now provided mostly by Classes DatagramSocket and DatagramPacket
- There is no real concept of a separate "server" sockets as there was with TCP sockets

UDP Support

- Objects are created of the Class DatagramSocket to act as endpoints for your communication

```
DatagramSocket mySocket = null;
mySocket = new DatagramSocket();
```

- can then use the send() and receive() methods of DatagramSocket

DatagramSocket's Constructors

- Has multiple constructors
- public **DatagramSocket**() throws SocketException
- public **DatagramSocket**(int port) throws SocketException
- public **DatagramSocket**(int port, InetAddress laddr) throws SocketException

DatagramSocket's Constructors

- public **DatagramSocket**() throws SocketException
- This creates a DatagramSocket bound to a port number chosen automatically by the system.
- This would be your normal action on the client side of an application

DatagramSocket's Constructors

- public **DatagramSocket**(int port) throws SocketException
- This creates a DatagramSocket bound to the port number that you specify.
- This would be the normal action at the server side of an application.
- Some systems may reject the request to use cerain port numbers, either because they are priviledged or perhaps because they are already in use.

DatagramSocket's Constructors

- public **DatagramSocket**(int port, InetAddress laddr) throws SocketException
- This creates a DatagramSocket bound to the specified port and on the specified Internet address.
- This would be used on the server side on machines with multiple interfaces.

DatagramSocket's Methods

- public void send(DatagramPacket p) throws IOException
- public void receive(DatagramPacket p) throws IOException

- These methods send and receive properly formatted UDP datagrams.

Further Methods to access information about the connection

- public void connect(InetAddress address, int port)
- public void disconnect()
- public InetAddress getInetAddress()
- public int getPort()
- public InetAddress getLocalAddress()
- public int getLocalPort()
- public void close()

Further Methods that specify non-blocking I/O

- public void setSoTimeout(int timeout) throws SocketException
- public int getSoTimeout() throws SocketException

Further Methods

- public void setSendBufferSize(int size) throws SocketException
- public int getSendBufferSize() throws SocketException
- public void setReceiveBufferSize(int size) throws SocketException
- public int getReceiveBufferSize() throws SocketException

DatagramPacket Class

- Used for both UDP datagrams to be sent and to store received UDP datagrams
- Because of the "connectionless" nature of UDP, the DatagramPackets contain the addressing information as well as application data
- Slightly different constructors are using for incoming and outgoing datagrams.

DatagramPacket Constructors

- public DatagramPacket(byte[] buf, int length)

used for incoming datagrams

- public DatagramPacket(byte[] buf, int length, InetAddress address, int port)

used for outgoing datagrams which will be sent to a "server" on the the Internet address and port number specified

DatagramPacket Constructors

- public DatagramPacket(byte[] buf,
  int offset, int length)

used for incoming datagrams when you want the data placed as some offset from the start of the buffer

- public DatagramPacket(byte[] buf, int offset, int length, InetAddress address, int port)

used for outgoing datagrams which will be sent to a "server" on the the Internet address and port number specified when you want the data taken from some offset from the start of the buffer

DatagramPacket Methods

- public InetAddress getAddress()
- public int getPort()
- public byte[] getData()
- public int getLength()

DatagramPacket Methods

- public int getOffset()
- public void setData(byte[] buf, int offset, int length)
- public void setAddress(InetAddress iaddr)
- public void setPort(int iport)
- public void setData(byte[] buf)
- public void setLength(int length)

Example Client

```
import java.io.*;
import java.net.*;

public class SendNote {
    public static void main(String[] args)
throws IOException {

  DatagramSocket ds = null;

  try {
     ds = new DatagramSocket();
  } catch (SocketException e) {
      System.err.println("SocketException");
      System.exit(1);
  }
```

```
 InetAddress ia =
      InetAddress.getByName("moin");
 BufferedReader br = new BufferedReader(
        new InputStreamReader(System.in));
 String poemLine;

while ( (poemLine = br.readLine()) != null) {
    byte[] outBuffer = poemLine.getBytes();
    DatagramPacket outPacket = new
        DatagramPacket(outBuffer,
                   outBuffer.length,
                   ia, 4200);
    ds.send(outPacket);
```

```
      /* Now get a reply */
      byte[] inBuffer = new byte[1024];
       DatagramPacket inPacket =
          new DatagramPacket(inBuffer,
                        inBuffer.length);
       ds.receive(inPacket);
       String s;
       s = new String(inPacket.getData(), 0,
                         inPacket.getLength());
       System.out.println("Reply was :" + s);
       }

       ds.close();
     }
   }
```

Sample Server

```
import java.io.*;
import java.net.*;

public class NoteServer{
  public static void main(String[] args) throws
IOException {

  DatagramSocket ds = null;

  try {
    ds = new DatagramSocket(4200);
  } catch (SocketException e) {
     System.err.println("SocketException on " +
                                   "port");
     System.exit(1);
  }
```

```
System.out.println("Have DatagramSocket about
to wait for Call");

DatagramPacket inPacket = null;
byte[] myBuffer;

myBuffer = new byte[4000];
inPacket = new DatagramPacket(myBuffer,
          myBuffer.length);
```

```
while (true) {
 try {
   inPacket.setLength(myBuffer.length);
   ds.receive(inPacket);
   String s;
   s = new String(inPacket.getData(), 0,
                       inPacket.getLength());
   System.out.println("I've been sent" + s);
```

```
  /* Now prepare the reply */
   DatagramPacket outPacket =
    new DatagramPacket(inPacket.getData(),
                       inPacket.getLength(),
                       inPacket.getAddress(),
                       inPacket.getPort() );
   ds.send(outPacket);
  } catch (IOException e) {
    System.err.println("incoming call failed");
  }
 }

/* ds.close(); */
}
}
```