Java UDP Client Server Program Example

Written by Nam Ha Minh
Last Updated on 18 July 2019 | Print M Email

In this Java Network programming tutorial, you will learn how to code a client/server application based on UDP protocol.

First, let's see how Java Network API is designed to support development of network applications that make use of UDP.

`DatagramPacket` and **`DatagramSocket`** are the two main classes that are used to implement a UDP client/server application. `DatagramPacket` is a data container and `DatagramSocket` is a mechanism to send and receive `DatagramPacket`s.

1. DatagramPacket

In UDP's terms, data transferred is encapsulated in a unit called datagram. A datagram is an independent, self-contained message sent over the network whose arrival, arrival time, and content are not guaranteed. And in Java, `DatagramPacket` represents a datagram.

You can create a `DatagramPacket` object by using one of the following constructors:

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

As you can see, the data must be in the form of an array of bytes. The first constructor is used to create a `DatagramPacket` to be received.

The second constructor creates a `DatagramPacket` to be sent, so you need to specify the address and port number of the destination host.

The parameter length specifies the amount of data in the byte array to be used, usually is the length of the array (`buf.length`).

There are also other constructors that allow you to specify the offset in the byte array, as well as using a `SocketAddress`:

- `DatagramPacket(byte[] buf, int offset, int length)`
- `DatagramPacket(byte[] buf, int offset, int length, SocketAddress address)`

In addition, the `DatagramPacket` provides setter and getter methods for address, data and port number. Consult its DatagramPacket Javadoc for full details.

2. DatagramSocket

You use `DatagramSocket` to send and receive `DatagramPacket`s. `DatagramSocket` represents a UDP connection between two computers in a network.

In Java, we use `DatagramSocket` for both client and server. There are no separate classes for client and server like TCP sockets.

So you create a `DatagramSocket` object to establish a UDP connection for sending and receiving datagram, by using one of the following constructors:

- `DatagramSocket()`
- `DatagramSocket(int port)`
- `DatagramSocket(int port, InetAddress laddr)`

The no-arg constructor is used to create a client that binds to an arbitrary port number. The second constructor is used to create a server that binds to the specific port number, so the clients know how to connect to.

And the third constructor binds the server to the specified IP address (in case the computer has multiple IP addresses).

These constructors can throw `SocketException` if the socket could not be opened, or the socket could not bind to the specified port or address. So you have catch or re-throw this checked exception.

The key methods of the `DatagramSocket` include:

- **`send`**`(DatagramPacket p)`: sends a datagram packet.
- **`receive`**`(DatagramPacket p)`: receives a datagram packet.
- **`setSoTimeout`**`(int timeout)`: sets timeout in milliseconds, limiting the waiting time when receiving data. If the timeout expires, a `SocketTimeoutException` is raised.
- **`close`**`()`: closes the socket.

These methods can throw `IOException`, `PortUnreachableException`, `SocketTimeoutException`... so you have to catch or re-throw them. Consult the `DatagramSocket` Javadoc for full details.

Now, let's see some sample programs in action.

3. Java UDP Client Example

We will write code for a client program that requests for quotes from a server that implements the Quote of the Day (QOTD) service - an Internet standard. The following code snippet sends a `DatagramPacket` to a server specified by hostname and port:

```
String hostname = "djxmmx.net";
int port = 17;

InetAddress address = InetAddress.getByName(hostname);
DatagramSocket socket = new DatagramSocket();

byte[] buffer = new byte[512];

DatagramPacket request = new DatagramPacket(buffer, buffer.length, address, port);
socket.send(request);
```

As you can see, the buffer has no data because QOTD service doesn't require a client sends any specific message. And you have to specify the server's address and port in the `DatagramPacket`. So this code just sends a signal to the server implying that "Hey, I'd like to get a quote from you".

And the following code snippet receives a `DatagramPacket` from the server:

```
DatagramPacket response = new DatagramPacket(buffer, buffer.length);
socket.receive(response);

String quote = new String(buffer, 0, response.getLength());

System.out.println(quote);
```

As you can see, once the socket is opened, receiving a packet is very simple. And the code converts the byte array to a String to be printed in readable format.

And here is the code of the full client program that parameterizes the hostname and port number, handles exceptions and gets a quote from the server for every 10 seconds:

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

/**
 * This program demonstrates how to implement a UDP client program.
 *
 *
 * @author www.codejava.net
 */
public class QuoteClient {

    public static void main(String[] args) {
        if (args.length < 2) {
            System.out.println("Syntax: QuoteClient <hostname> <port>");
            return;
        }

        String hostname = args[0];
        int port = Integer.parseInt(args[1]);

        try {
            InetAddress address = InetAddress.getByName(hostname);
            DatagramSocket socket = new DatagramSocket();

            while (true) {

                DatagramPacket request = new DatagramPacket(new byte[1], 1, address, port);
                socket.send(request);

                byte[] buffer = new byte[512];
                DatagramPacket response = new DatagramPacket(buffer, buffer.length);
                socket.receive(response);

                String quote = new String(buffer, 0, response.getLength());

                System.out.println(quote);
                System.out.println();

                Thread.sleep(10000);
            }

        } catch (SocketTimeoutException ex) {
            System.out.println("Timeout error: " + ex.getMessage());
            ex.printStackTrace();
        } catch (IOException ex) {
            System.out.println("Client error: " + ex.getMessage());
            ex.printStackTrace();
        } catch (InterruptedException ex) {
            ex.printStackTrace();
        }
    }
}
```

To test this client program, type the following command:

```
java QuoteClient djxmmx.net 17
```

`djxmmx.net` is a public QOTD server we can use, and 17 is the port number reserved for QOTD service.

You would see the output something like this:

```
"When a stupid man is doing something he is ashamed of, he always declares that it is his duty." George Bernard Shaw (1856-1950)

"Oh the nerves, the nerves; the mysteries of this machine called man!
 Oh the little that unhinges it, poor creatures that we are!"
 Charles Dickens (1812-70)
```

```
6
7   In Heaven an angel is nobody in particular." George Bernard Shaw (1856-1950)
```

If you test this program yourself, you may see different quotes because the server returns random quotes. Press Ctrl + C to terminate the program.

4. Java UDP Server Example

The following sample program demonstrates how to implement a server for the above client. The following code creates a UDP server listening on port 17 and waiting for client's request:

```
1  DatagramSocket socket = new DatagramSocket(17);
2
3  byte[] buffer = new byte[256];
4
5  DatagramPacket request = new DatagramPacket(buffer, buffer.length);
6  socket.receive(request);
```

The `receive()` method blocks until a datagram is received. And the following code sends a `DatagramPacket` to the client:

```
1  InetAddress clientAddress = request.getAddress();
2  int clientPort = request.getPort();
3
4  String data = "Message from server";
5  buffer = data.getBytes();
6
7  DatagramPacket response = new DatagramPacket(buffer, buffer.length, clientAddress, clientPort);
8  socket.send(response);
```

As you can see, the server also needs to know client's address and port to send the `DatagramPacket`. This information can be obtained from the `DatagramPacket` received from the client previously. And a String is converted to an array of bytes which then can be wrapped in a `DatagramPacket`.

And the following is a full-featured server program that reads quotes from a text file, and sends a random quote for every client's request. The quote file and port number are given as program's arguments. Here's the code:

```
1  import java.io.*;
2  import java.net.*;
3  import java.util.*;
4
5  /**
6   * This program demonstrates how to implement a UDP server program.
7   *
8   *
9   * @author www.codejava.net
10  */
11 public class QuoteServer {
12     private DatagramSocket socket;
13     private List<String> listQuotes = new ArrayList<String>();
14     private Random random;
15
16     public QuoteServer(int port) throws SocketException {
17         socket = new DatagramSocket(port);
18         random = new Random();
19     }
20
21     public static void main(String[] args) {
22         if (args.length < 2) {
23             System.out.println("Syntax: QuoteServer <file> <port>");
24             return;
25         }
26
27         String quoteFile = args[0];
28         int port = Integer.parseInt(args[1]);
29
30         try {
31             QuoteServer server = new QuoteServer(port);
32             server.loadQuotesFromFile(quoteFile);
33             server.service();
34         } catch (SocketException ex) {
35             System.out.println("Socket error: " + ex.getMessage());
36         } catch (IOException ex) {
37             System.out.println("I/O error: " + ex.getMessage());
38         }
39     }
40
41     private void service() throws IOException {
42         while (true) {
43             DatagramPacket request = new DatagramPacket(new byte[1], 1);
44             socket.receive(request);
45
46             String quote = getRandomQuote();
47             byte[] buffer = quote.getBytes();
48
49             InetAddress clientAddress = request.getAddress();
50             int clientPort = request.getPort();
51
52             DatagramPacket response = new DatagramPacket(buffer, buffer.length, clientAddress, clientPort);
53             socket.send(response);
54         }
55     }
56
57     private void loadQuotesFromFile(String quoteFile) throws IOException {
58         BufferedReader reader = new BufferedReader(new FileReader(quoteFile));
59         String aQuote;
60
61         while ((aQuote = reader.readLine()) != null) {
62             listQuotes.add(aQuote);
63         }
64
65         reader.close();
66     }
67
68     private String getRandomQuote() {
69         int randomIndex = random.nextInt(listQuotes.size());
70         String randomQuote = listQuotes.get(randomIndex);
71         return randomQuote;
72     }
```

```
73 }
```

Suppose we have a `Quotes.txt` file with the following content (each quote is in a single line):

```
Whether you think you can or you think you can't, you're right - Henry Ford
There are no traffic jams along the extra mile - Roger Staubach
Build your own dreams, or someone else will hire you to build theirs - Farrah Gray
What you do today can improve all your tomorrows - Ralph Marston
Remember that not getting what you want is sometimes a wonderful stroke of luck - Dalai Lama
```

Type the following command to run the server program:

```
java QuoteServer Quotes.txt 17
```

And run the client program (on the same computer):

```
java QuoteClient localhost 17
```

Both the client and server are running in an infinite loop, so you have to press Ctrl + C to terminate.

That's the lesson about how to develop a network client/server application relying on UDP protocol. Based on this knowledge, you are able to develop client programs that communicate with servers via UDP, and developing your own UDP client/server applications.

API References:

- DatagramPacket Javadoc
- DatagramSocket Javadoc

Related Java Network Tutorials:

- Java InetAddress Examples
- Java Socket Client Examples (TCP/IP)
- Java Socket Server Examples (TCP/IP)

Other Java network tutorials:

- How to use Java URLConnection and HttpURLConnection
- Java URLConnection and HttpURLConnection Examples
- Java HttpURLConnection to download file from an HTTP URL
- Java HTTP utility class to send GET/POST request
- How to Create a Chat Console Application in Java using Socket
- Java upload files by sending multipart request programmatically

About the Author:

Nam Ha Minh is certified Java programmer (SCJP and SCWCD). He began programming with Java back in the days of Java 1.4 and has been passionate about it ever since. You can connect with him on Facebook and watch his Java videos on YouTube.