

CSCE 515 Computer Network Programming

Project 4, Due Date: December 4, 11:59pm, 2008

In this project, you are required to write the simple multicast chat system using JAVA. You need to write a centralized chat server and a chat client. The system should use a message passing system over sockets to allow multiple clients to chat with each other. There should be a central server that clients use to log into the system. From that point, the user should be able to chat with all other clients signed into the chat server.

You need to write a network-client that contacts your server, and lets the server know when the user logs in and out of the system. More specifically, your network-client needs to achieve the following requirements:

1. (1 pts) The network-client should contact the server in the Constructor. Assume the hostname and the port number of the server are well-known. (*You should either use `apollon.cse.sc.edu` or `broad.cse.sc.edu` to host the server*) Your client will first prompt a welcome message that asks the user to enter a username using the keyboard. This username will then be sent to the server. Then, your server, after receiving the username from your client, will send an acknowledgment message and the multicast group address to the client.
2. (2 pt) Your client, after receiving the acknowledgment message and the multicast group address from your server, will join the multicast group and prompt a message to ask the user to enter the text message to be sent to other clients. Each time the user hit 'enter', the client sends the message line to the server.
3. (2 pt) From this point on, the client will send (unicast) all message entered by the user to the server, and receive and display all messages received from the multicast chat group and the server. The client should display the name of the user that sends the messages to the server and the message itself.
4. (1 pts) To exit the client, a user should input a line with '.' only. Before the client can exit, the client should send sign off message to the server, leave the multicast chat group, and wait for the confirmation message from the server.

You need to write a server. This server must keep track of all users in your chat system. The server should be run on either `apollon.cse.sc.edu` or `broad.cse.sc.edu`. It should operate on a known port. You should choose a port and make sure that your client will connect to the right port.

1. (3 pt) The network-server should listen on a well-know port and accept user's sign in request. The server should send back a welcome message (including the multicast group number) when a user first joins the chat group, and the server should also inform everybody who is already in the group that a new user "bla" has just joined the chat group. The server keeps the list of users that is currently signed in.
2. (3 pts) Your server should waiting for the client to send message. Each time a new message is received, the server add the user's name to the message and send it out to the multicast group.
3. (1 pt) When a server received a user 'bla' sign-off message, the server will send back the confirmation to the user 'bla', and inform all other clients that user 'bla' has just left the chat group.

You need to design your chat message format, including the sign-in message, sign-off message and regular message.

This project will count for 15 points toward your final grade. The programs will be graded based on the amount of the required functionalities that has been implemented (13 points), the quality of the code (2 points). You need to submit all the source code necessary to build your system, a Makefile, a README file and your report via the Dropbox named project04 before the due day. The README file should contain following information: 0) your name (graduate or undergraduate); 1) the programming environment you used; 2) how to compile your programs; 3) how to execute your programs; which machine should host your server. 4) the name of each file submitted along with a one line description of what is in the file; 5) If your code is not complete, tell me what works and what doesn't.

To start:

Check the sample code:

MulticastSender.java

MulticastSniffer.java

First, start the receiver:

```
Session1 % java MulticastSniffer all-systems.mcast.net 4000
```

Then start a Sender:

```
Session2 % java MulticastSender all-systems.mcast.net 4000
```

What you should get at session 1:

Here's some multicast data

Here's some multicast data

Here's some multicast data

Here's some multicast data

Here's some multicast data

Here's some multicast data

Here's some multicast data

Here's some multicast data

Here's some multicast data

```

// MulticastSender.java
import java.io.*;
import java.net.*;

public class MulticastSender {

    public static void main(String[] args) {

        InetAddress ia = null;
        int port = 0;
        String characters = "Here's some multicast data\n";
        byte[] data = new byte[characters.length()];

        // read the address from the command line
        try {
            ia = InetAddress.getByName(args[0]);
            port = Integer.parseInt(args[1]);
        } catch (Exception e) {
            System.err.println(e);
            System.err.println("Usage: java MulticastSender MulticastAddress port");
            System.exit(1);
        }

        characters.getBytes(0, characters.length(), data, 0);
        DatagramPacket dp = new DatagramPacket(data, data.length, ia, port);

        try {
            MulticastSocket ms = new MulticastSocket();
            ms.joinGroup(ia);
            for (int i = 1; i < 10; i++) {
                ms.send(dp, (byte) 1);
            }
            ms.leaveGroup(ia);
            ms.close();
        } catch (SocketException se) {
            System.err.println(se);
        } catch (IOException ie) {
            System.err.println(ie);
        }

    }

}

```

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

public class MulticastSniffer {

    public static void main(String[] args) {

        InetAddress ia = null;
        byte[] buffer = new byte[65509];
        int port = 0;

        // read the address from the command line
        try {
            ia = InetAddress.getByName(args[0]);
            port = Integer.parseInt(args[1]);
        } // end try
        catch (Exception e) {
            System.err.println(e);
            System.err.println("Usage: java MulticastSniffer MulticastAddress port");
            System.exit(1);
        }

        try {
            MulticastSocket ms = new MulticastSocket(port);
            ms.joinGroup(ia);
            while (true) {
                DatagramPacket dp = new DatagramPacket(buffer, buffer.length);
                ms.receive(dp);
                String s = new String(dp.getData(), 0, 0, dp.getLength());
                System.out.println(s);
            }
        }
        catch (SocketException se) {
            System.err.println(se);
        }
        catch (IOException ie) {
            System.err.println(ie);
        }
    }
}
```