CS4133/5133: Data Networks

Project-3

Multicast System to UDP

Fall 2017

100 pts

Due Date: Nov 05, 2017

1. Objective:

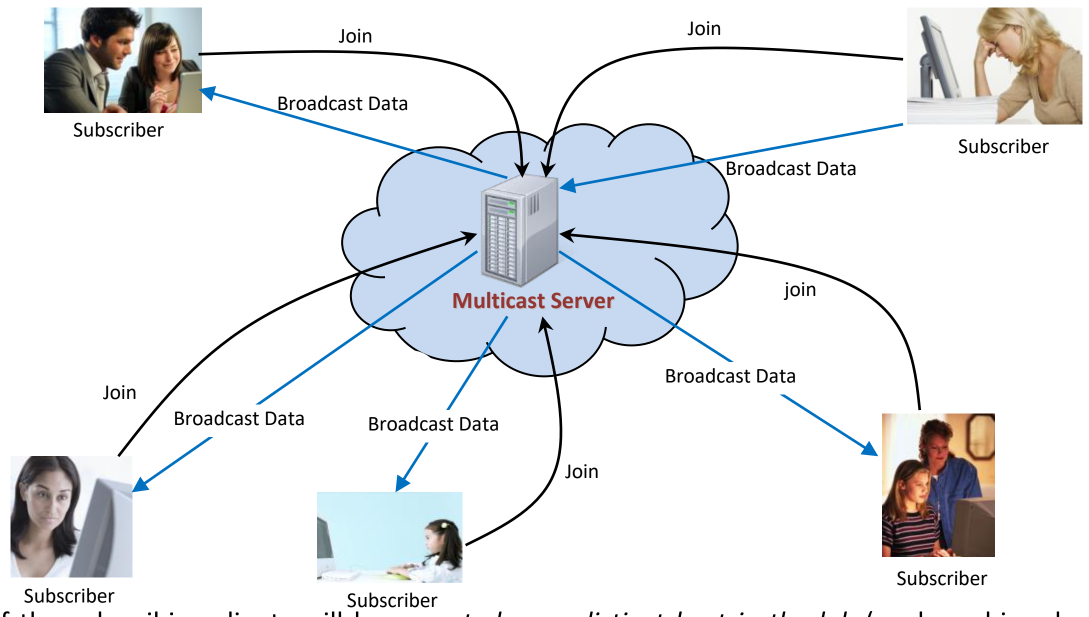
The objective of this third programming project is to learn UDP client-server interaction using UDP socket interface in C programming language.

2. Project Specification:

Overall System Behavior:

Multicast only applies to UDP, which is connectionless and provide no reliability. Therefore, multicast is also unreliable as there is no guarantee that every multicast message destined for the same multicast group address will be received by every host subscribing to this multicast group. In order to fit the reliability needs of multicast applications, many schemes have been proposed. For more information about reliable multicast protocol, you can check the web page of Reliable Multicast Transport Protocol (RMTP) at http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.83.4272&rep=rep1&type=pdf.

In this project, you are required to implement a Multicast Server and a Client in C language that implements a simplest version of multicasting. Allow up to 5 clients to simultaneously subscribe to the multicast group served by your Multicast Server.



Each of the subscribing clients will be *executed on a distinct host in the lab* (gpel machine, however, since our goal is to learn only, you may execute from the same machine), and the multicast server will keep track of all current subscribed clients. Every time the server multicasts a message.

2.1. Multi-cast Server Behavior:

- At startup, the server takes an argument that specifies the port number that it is listening to. You have to use (*5000+last 4 digits* of your student-id number) to avoid requesting same port by multiple students.
- After successful startup, the multi-cast server program will ask the user to create a group (e.g., OUCS etc.) and maximum number of clients (e.g., 5) can join in the group to receive broadcast messages.
- Next, your multicast server should prompt the user to enter a number of messages to be multi-casted among clients. At the same time, your multicast server should listen for joining and leaving requests from the clients, until the user finishes inputting all messages.
- As clients may use the message "JOIN" to join the multicast group and "QUIT" to leave from the group, server needs to maintain the active client list for message broadcasting purpose. If the client is unable to join the multicast group, send an error Message to client (e.g., more than maximum number of clients join the multicast group). If any client is removed from the active client list, the server should print out the client IP + port information.
- When sending out a message, your multicast server should display the content of the message.
- When accepting a message, your multicast client should display the content of the message.
- Server requires unsubscribing all clients from the group (e.g., OUCS) as soon as it receives the string "CLEARALL" from user input.

2.2. Client Behavior:

- Your client program needs to take two arguments from user during startup that specifies the IP address and port number of the multi-cast server it wants to access.
- After a successful startup, your sender program will first prompt a welcome message that asks the user to input a group name (e.g., OUCS etc.) it wants to join.
- Next, client requires to send the message "JOIN" to join the multicast group to receive messages from the multi-cast server.
- Similarly, client can send the message "QUIT" to unsubscribe from the group and receive no further messages from the server.
- When accepting a message, your multicast client should display the content of the message.
- In order to simulate possible message loss, your clients should "accept" only 90% received messages and discard the remaining 10% [This part is Optional, If you implement it, then mention in the documentation, if you don't implement, also mention it in the documentation, no marks will be deducted].

3. Programming Notes:

You are required to work on this project alone. You have to use UDP sockets for implementing both client and server programs. You are allowed to use multi-threading in this project, if requires. You should program your each process to print an informative statement whenever it takes an action (e.g., sends or receives a message, detects termination of input, etc), so that you can see that your processes are working correctly (or not!). One should be able to determine from this output if your processes are working correctly. You should hand in screen shots (or file content, if your process is writing to a file) of these informative messages.

Make sure, your multicast server allows the user to specify the listening port number and multicast group name during startup. Make sure you close every socket that you use in your program. If you abort your program, the socket may still hang around and the next time you try and bind a new socket to the port ID you previously used (but never closed), you may get an error. Also, please be aware that port ID's, when bound to sockets, are system-wide values and thus other students may be using the port number you are trying to use. If you need to kill a process after you have started it, you can use the UNIX kill command, use the UNIX ps command to find the process id of your server.

Many of you will be running the client and multi-cast server on the same UNIX machine (e.g., by starting up the multicast-server in the background, then starting up the client). This is fine; since you are using UDP sockets for communication, these processes can run on the same machine or different machines without modification.

Any clarifications and revisions to the project will be posted on the course web page on Canvas (http://canvas.ou.edu). Students are encouraged to start this project early and use the discussion list on Canvas for any general question so that everyone can benefit from the replies. Please do not use the list to flame others, and under no circumstances to post solutions or hints to solutions for others.

4. File names:

File names for this project are as follows:

Client: *lastNameUDPClient.c*

Multi-cast Server: lastNameUDPServer.c

5. Points Distribution:

Bits and pieces	Points
Multicast Client Program	30
Multicast Server Program	50
Program Style (Coding style, comments etc.)	10
Documentation	10

6. Submission Instructions:

This project requires the submission of a *soft copy* and a *hard copy. Plagiarism* will not be tolerated under any circumstances. Participating students will be penalized depending on the degree of plagiarism.

6.1. Soft Copy (Due November 05, 2017 , 11:59 pm)

The soft copy should consist of:

- source code of the Client program,
- source code of the Multicast server program,
- any header file(s), and
- detailed documentation should consist of following sections:
    - discussion of your problem-solving approach,
    - detailed analysis of your implemented codes, and
    - screen shots of outputs

These must be submitted through Canvas (http://canvas.ou.edu).

6.2. Hard copy (Due November 06, 2017, beginning of the class)

The documentation, submitted as soft copy, should be submitted at the beginning of the class. The hard copy must be the same as the soft copy.

Please **DO NOT MODIFY** the project code in any way after the deadline of soft copy submission.

7. Late Penalty:

Submit your project on or before the due date to avoid any late penalty. **A late penalty of *15% per day* will be imposed strictly after the *due date***. After one week from the due date, you will not be allowed to submit the project.

Good Luck!!