Lab 4: UDP Multicast

**Purpose**

This laboratory work will introduce UDP multicast. You will code a simple client-server program that multicasts a given message from a single client to multiple servers.

**References**

Read an overview on Multicast addresses: [http://www.tcpipguide.com/free/t\_IPMulticastAddressing.htm (Links to an external site.)](http://www.tcpipguide.com/free/t_IPMulticastAddressing.htm)

Wikipedia also offers decent introductions:

[https://en.wikipedia.org/wiki/IP\_multicast (Links to an external site.)](https://en.wikipedia.org/wiki/IP_multicast)

[https://en.wikipedia.org/wiki/Multicast\_address (Links to an external site.)](https://en.wikipedia.org/wiki/Multicast_address)

**UDP Multicast Class**

You will use the UdpMulticast class found on canvas. See these files:  [UdpMulticast.cpp](https://canvas.uw.edu/courses/1465757/files/75458235/download?wrap=1)[download](https://canvas.uw.edu/courses/1465757/files/75458235/download?download_frd=1)[UdpMulticast.h](https://canvas.uw.edu/courses/1465757/files/75458260/download?wrap=1)[download](https://canvas.uw.edu/courses/1465757/files/75458260/download?download_frd=1)

A description of the class is found in the following table:

|  |  |
| --- | --- |
| **Methods** | **Descriptions** |
| UdpSocket(char group[], int port) | Constructor opens a UDP datagram socket and has it participate in a given group address (e.g. "238.255.255.255") at a given port. |
| ~UdpSocket() | Destructor which closes the UDP socket. |
| int getClientSocket() | uses this socket descriptor as a client multicast socket. It returns a socket descriptor but you do not directly use the descriptor. |
| int getServerSocket() | uses this socket descriptor as a server multicast socket. It returns a socket descriptor but you do not directly use the descriptor. |
| bool multicast(char buf[]) | has a client multicast a give message in buf[] to all servers belonging to the same group@port. It returns true upon a success, otherwise false. |
| recv(char buf[], int size) | has a server receive a multicast message into a given buf[] with the size. It returns true upon a success, otherwise false. |

**Statement of Work**

The following code is a template for launching a UDP Multicast client and a server. It reads three arguments: a group address (e.g., 238.255.255.255) in argv[0], a port (e.g., 5001) in argv[1], and a message in argv[2].

If a message is not given (only two arguments given), the program behaves as a server, otherwise it behaves as a client. Find below some starter code. This code is also posted on canvas: [lab4\_temp.cpp](https://canvas.uw.edu/courses/1465757/files/75458267/download?wrap=1)[download](https://canvas.uw.edu/courses/1465757/files/75458267/download?download_frd=1)

#include "UdpMulticast.h"

#include <iostream>

using namespace std;

#define SIZE 1024

int main(int argc, char \*argv[])

{

   // validate arguments

   if (argc < 3)

   {

      cerr << "usage: lab4 group port [message]" << endl;

      return -1;

   }

   char \*group = argv[1];

   int port = atoi(argv[2]);

   if (port < 5001)

   {

      cerr << "usage: lab4 group port [message]" << endl;

      return -1;

   }

   char \*message = (argc == 4) ? argv[3] : NULL;

   // if message is null, the program should behave as a server,

   // otherwise as a client.

   UdpMulticast udp(group, port);

   if (message != NULL)

   {

      // client

      // implement yourself

   }

   else

   {

      udp.getServerSocket();

      while (true)

      {

         // server

         // implement yourself

      }

   }

return 0;

}

Complete the main() function and compile the program as follows:

css503@csslab3 lab4]$ g++ lab4.cpp udp\_multicast.cpp -o lab4

Test your program on two machines in the linux lab:

*Start the server first on one machine:*

[css503@csslab3 lab4]$ ./lab4 238.255.255.255 5001

*In another machine, run the client:*

[css503@csslab5 lab4]$ ./lab4 238.255.255.255 5001 Hello!   
[css503@csslab5 lab4]$ ./lab4 238.255.255.255 5001 Bye!   
[css503@csslab5 lab4]$

*Look at the server behavior:*

[css503@csslab3 lab4]$ ./lab4 238.255.255.255 5001  
   Hello!   
   Bye!

**What to Turn in**

As this is a lab you are required to work on this with one other student in the class.  Make sure you turn in only one submission for both students but make sure both names are on the assignment.

* Lab4.cpp
* Execution output