**Institute of Technology, Tralee**

**Computing Department**

Distributed Computing

Group Communication

Do the following exercises and write the answers into your log book.

# Exercise 1:

The following code samples show the coding for a simple example multicast application when run each receiver process subscribes to the multicast group and listens for a message. The sender process is not a member of the multicast group it sends a single message to the multicast before closing its multicast socket.

Compile the Example1.java program, then execute it in each of the following sequences, describe and explain the outcome of each:

1. Start two or more Receiver processes, and then start a sender process with a message of your choice. Describe and explain the outcome
2. Start a sender process with a message of your choice first, then two or more receiver processes. Describe and explain the outcome.

Example1Sender

import java.io.\*;

import java.net.\*;

/\*\*

\* This example illustrates the basic syntax for basic multicast.

\* @author M. L. Liu

\*/

public class Example1Sender {

// An application which uses a multicast socket to send

// a single message to a multicast group.

// The message is specified as a command-line argument

public static void main(String[] args) {

MulticastSocket s;

InetAddress group;

if (args.length != 1)

System.out.println

("This program requires a command line argument");

else {

try {

// create the multicast socket

group = InetAddress.getByName("239.1.2.3");

s = new MulticastSocket(3456);

s.setTimeToLive(32); // restrict multicast to processes

// running on hosts at the same site.

String msg = args[0];

DatagramPacket packet =

new DatagramPacket(msg.getBytes(), msg.length(),

group, 3456);

s.send(packet);

s.close();

}

catch (Exception ex) { // here if an error has occurred

ex.printStackTrace( );

} // end catch

}//end else

}// end main

}// end class

Example1Receiver

import java.io.\*;

import java.net.\*;

/\*\*

\* This example illustrates the basic syntax for basic multicast.

\* @author M. L. Liu

\*/

public class Example1Receiver {

// An application which joins a multicast group and

// receives a single message sent to the group.

static MulticastSocket s;

static InetAddress group;

public static void main(String[] args) {

try {

// join a Multicast group and send the group salutations

group = InetAddress.getByName("239.1.2.3");

s = new MulticastSocket(3456);

System.out.println("Joined group at 239.1.2.3 port 3456");

s.joinGroup(group);

byte[] buf = new byte[100];

DatagramPacket recv = new DatagramPacket(buf, buf.length);

s.receive(recv);

System.out.println(new String(buf));

s.close();

}

catch (Exception ex) { // here if an error has occurred

ex.printStackTrace( );

}

}// end main

}// end class

# Exercise 2

In this example each process in the multicast group sends a message and each process also displays all of the messages it receives as a member of the multicast group. In the example a thread is spawned to receive and display the messages. To ensure that each process is ready to receive a pause is imposed before the process is allowed to send its message.

Compile the Example2.java program, then execute it in each of the following sequences, describe and explain the outcome of each:

1. Start two or more Receiver processes, and then start a sender process with a message of your choice. Describe and explain the outcome
2. Start a sender process with a message of your choice first, then two or more receiver processes. Describe and explain the outcome.
3. Try to send messages to all the machines in the lab – what do you need to do?

Example2SenderReceiver

// This program illustrates sending and receiving using mutlicast

import java.io.\*;

import java.net.\*;

/\*\*

\* This example illustrates using multithreads to send and

\* receive multicast in one process.

\* @author M. L. Liu

\*/

public class Example2SenderReceiver{

// An application which uses a multicast socket to send

// a single message to a multicast group, and a separate

// thread which uses a separate multicast socket to receive

// messages sent to the same group.

// Three command-line arguments are expected:

// <multicast IP address>,<multicast port>,<message>

public static void main(String[] args) {

InetAddress group = null;

int port = 0;

MulticastSocket socket = null;

String characters;

byte[] data = null;

if (args.length !=3)

System.out.println("Three command-line arguments are expected.");

else {

try {

group = InetAddress.getByName(args[0]);

port = Integer.parseInt(args[1]);

characters = args[2];

data = characters.getBytes();

DatagramPacket packet =

new DatagramPacket(data, data.length, group, port);

Thread theThread =

new Thread(new ReadThread(group, port));

theThread.start();

System.out.println("Hit return when ready to send:");

InputStreamReader is = new InputStreamReader(System.in);

BufferedReader br = new BufferedReader(is);

br.readLine();

socket = new MulticastSocket(port);

socket.setTimeToLive(1);

socket.send(packet);

socket.close();

}

catch (Exception se) {

se.printStackTrace( );

} // end catch

} //end else

} // end main

} // end class

ReadThread

import java.net.\*;

import java.io.\*;

/\*\*

\* This class is to be used with Example2SenderReceiver for

\* reading multicast messages while the main thread sends

\* a multicast message. Each message read is echoed on the

\* screen.

\* @author M. L. Liu

\*/

class ReadThread implements Runnable {

static final int MAX\_LEN = 30;

private InetAddress group;

private int port;

public ReadThread(InetAddress group, int port) {

this.group = group ;

this.port = port;

}

public void run() {

try {

MulticastSocket socket = new MulticastSocket(port);

socket.joinGroup(group);

while (true) {

byte[] data = new byte[MAX\_LEN];

DatagramPacket packet =

new DatagramPacket(data, data.length, group, port);

socket.receive(packet);

String s = new String(packet.getData());

System.out.println(s);

} //end while

} // end catch

catch (Exception exception) {

exception.printStackTrace( );

}

} // end run

} //end class

# Exercise 3:

Based on Example1Receiver.java, create a program Example1aReceiver.java which joins a multicast group of a different IP address (e.g. 239.1.2.4) but the same port. Compile Example1aReceiver.java start two or more Example1Receiver processes first then a Example1aReceiver processes and then a sender process with a message of your choice. Does the Example1aReceiver process receive the message? Describe and explain the outcome.