**1. Socket**

**My client.java**

import java.net.\*;

import java.io.\*;

public class MyClient {

public static void main(String[] args) throws Exception{

//The socket object takes ip and port number of the server which client wants to connect

Socket s = new Socket("127.0.0.1",5555);

System.out.println("Connected to Server, Please type your message and hit Enter to send");

//Reading input from KeyBoard

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

//OutputStream object to write to Server

OutputStream ostream = s.getOutputStream();

//PrintWriter object to send the data to the outputstream

PrintWriter pw = new PrintWriter(ostream, true);

//InputStream objects to recieve from Server

InputStream istream = s.getInputStream();

//Reading receieved message from Server

BufferedReader recieve = new BufferedReader(new InputStreamReader(istream));

//Client Message and Server Message objects

String clientmessage = "";

String servermessage = "";

while(true)

{

//Input Message to be sent to Server

System.out.print("Client: ");

clientmessage = br.readLine();

//print writer object sending the message to the socket through outputstream

pw.println(clientmessage);

//if the message is bye end the communication here

if(clientmessage.equals("bye"))

{

break;

}

//Read the inputstream of the server from the socket

servermessage = recieve.readLine();

System.out.println("Server: "+servermessage);

//if the message is bye end the communication here

if(servermessage.equals("bye"))

{

break;

}

}

//closing all the streams and sockets

s.close();

istream.close();

ostream.close();

System.out.println("Connection Terminated");

}

}

**My server.java**

/\*

Name: Sahil A. Shaikh

Roll no: 43365

Problem Statement: To develop any distributed application through implementing client-server communication programs based on Java Sockets.

\*/

import java.net.\*;

import java.io.\*;

public class MyServer {

public static void main(String[] args) throws Exception{

//Creating a port for communication

ServerSocket ss = new ServerSocket(5555);

System.out.println("Server Initiated, Waiting for Client to Connect...");

//Binding Client and Server on port 5555

Socket s = ss.accept();

System.out.println("Client Connected");

//Reading input from KeyBoard

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

//OutputStream object to write to clients

OutputStream ostream = s.getOutputStream();

//PrintWriter object to send the data to the outputstream

PrintWriter pw = new PrintWriter(ostream,true);

//InputStream objects to recieve from Client

InputStream istream = s.getInputStream();

//Reading receieved message from client

BufferedReader recieve = new BufferedReader(new InputStreamReader(istream));

//Client Message and Server Message objects

String servermessage = "";

String clientmessage = "";

while(true)

{

//Read the inputstream of the client from the socket

clientmessage = recieve.readLine();

System.out.println("Client: "+clientmessage);

//if the message is bye end the communication here

if(clientmessage.equals("bye"))

{

break;

}

//Server writing its message

System.out.print("Server: ");

servermessage = br.readLine();

//print writer object sending the message to the socket through outputstream

pw.println(servermessage);

if(servermessage.equals("bye"))

{

break;

}

}

//closing all the streams and sockets

s.close();

ss.close();

istream.close();

ostream.close();

System.out.println("Connection Terminated");

}

}

**Commands:**

**1st cmd**

Javac myserver.java

Java myserver

2nd cmd

Javac myclient.java

Java myclient

**2.rmi**

**Client.java**

import java.rmi.\*;

import java.util.Scanner;

public class Client {

public static void main(String args[]) {

try {

Scanner s = new Scanner(System.in);

System.out.println("Enter the Server address : ");

String server = s.nextLine();

ServerInterface si = (ServerInterface) Naming.lookup("rmi://" + server + "/Server");

System.out.println("Enter first string : ");

String first = s.nextLine();

System.out.println("Enter second string : ");

String second = s.nextLine();

System.out.println("Concatenated String : " + si.concat(first, second));

s.close();

} catch (Exception e) {

System.out.println(e);

}

}

}

**Servant.java**

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

import java.rmi.\*;

import java.rmi.server.\*;

public class Servant extends UnicastRemoteObject implements ServerInterface {

protected Servant() throws RemoteException {

super();

}

@Override

public String concat(String a, String b) throws RemoteException {

return a + b;

}

}

**Server.java**

import java.rmi.\*;

import java.net.\*;

public class Server {

public static void main(String[] args) {

try {

Servant s = new Servant();

Naming.rebind("Server", s);

} catch (Exception e) {

System.out.println(e);

}

}

}

**Commands:**

Javac \*.java

Rmic Servant

Start rmregistery

Java Server

Java Client

Server address: 127.0.0.1

**3.corba**

**Client.java**

import HelloModule.\*;

import org.omg.CosNaming.\*;

import org.omg.CosNaming.NamingContextPackage.\*;

import org.omg.CORBA.\*;

import org.omg.CORBA.ORB.\*;

import java.util.Scanner;

public class Client {

public static void main(String[] args) {

Hello HelloImpl = null;

try {

// create and initialize ORB

org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);

//obtaining the ORB object references for initial services

org.omg.CORBA.Object objRef = orb.resolve\_initial\_references("NameService");

//Naming ContextExt contains set of name bindings of Interoperable Naming services

NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

//We have binded the name Hello from server so using same name for lookup

String name = "Hello";

//Getting reference of server name hello and then we are narrowing it down to Hello type

HelloImpl = HelloHelper.narrow(ncRef.resolve\_str(name));

//Taking user Input

System.out.println("Enter your name: ");

Scanner sc = new Scanner(System.in);

String userName = sc.nextLine();

//Invoking the print\_hello

System.out.println(HelloImpl.print\_hello(userName));

} catch (Exception e) {

System.out.println(e);

}

}

}

**Server.java**

import HelloModule.Hello;

import org.omg.CosNaming.\*;

import org.omg.CosNaming.NamingContextPackage.\*;

import org.omg.CORBA.\*;

import org.omg.PortableServer.\*;

public class Server {

public static void main(String[] args) {

try {

// create and initialize ORB

org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);

//Getting reference of ROOTPOA

POA rootPOA = POAHelper.narrow(orb.resolve\_initial\_references("RootPOA"));

//Activating ROOTPOA

rootPOA.the\_POAManager().activate();

//Create Object of Interface implementation which will act as servant

HelloImpl helloImpl = new HelloImpl();

//Registering the servant object reference in the rootPOA

org.omg.CORBA.Object ref = rootPOA.servant\_to\_reference(helloImpl);

//narrowing the ROOTPOA reference object to propertype which in this case is of type Hello

System.out.println("Step 1");

Hello h\_ref = HelloModule.HelloHelper.narrow(ref);

//obtaining the ORB object references for initial services

System.out.println("Step 2");

org.omg.CORBA.Object objRef = orb.resolve\_initial\_references("NameService");

//Afain narrowing the ORB object reference to NamingContext type to bin it with server

System.out.println("Step 3");

NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

//passing path and the servant object to the naming service, binding the servant object to the "Hello"

System.out.println("Step 4");

String name = "Hello";

NameComponent path[] = ncRef.to\_name(name);

ncRef.rebind(path,h\_ref);

//Enbaling ORB to run on main thread and waiting till invocation comes for ORB. Since it is in main method after invocation it will wait again

System.out.println("Server Ready....");

orb.run();

} catch (Exception e) {

System.out.println(e);

}

}

}

**Helloimpl.java**

import HelloModule.HelloPOA;

class HelloImpl extends HelloPOA{

HelloImpl()

{

super();

System.out.println("Ready");

}

public String print\_hello(String s)

{

return("Hello "+s);

}

}

**Commands:**

idlj -fall Hello.idl

javac *\*.java HelloModule/\**.java

orbd -ORBInitialPort 1050&

server code-- java Server -ORBInitialPort 1050& -ORBInitialHost localhost&

client code-- java ReverseClient -ORBInitialPort 1050 -ORBInitialHost localhost

**4.rmi**

Installation of OPENMPI

1. Download openmpi-4.1.4.tar.bz2 from http://www.open-mpi.org in a folder say LP5.
2. Goto the terminal (Command prompt) 3. update using

sudo apt-get update

sudo apt install gcc {if not already installed}

4. Goto the directory which contains the downloaded file 5. Extract the files using

tar -jxf openmpi-4.1.4.tar.bz2

1. The directory openmpi-4.1.4 is created
2. Configure, compile and install by executing the following commands

./configure --prefix=$HOME/opt/openmpi

make all make install

1. Now openmpi folder is created in ‘opt‘ folder of Home directory.
2. Now the folder LP5 can be deleted (optional)
3. Update the PATH and LD\_LIBRARY\_PATH environment variable using

echo "export PATH=\$PATH:\$HOME/opt/openmpi/bin" >> $HOME/.bashrc

echo "export LD\_LIBRARY\_PATH=\$LD\_LIBRARY\_PATH:\$HOME/opt/openmpi/lib">>$HOME/.bashrc

1. Compile the program using

# mpicc name of the program

12. Execute the program using

mpirun -np N ./a.out

Hello world program

nllabc2d22@nllabc2d-22:~/opt/openmpi/bin$ gedit hello.c

#include <stdio.h> #include "mpi.h" int main(int argc, char\* argv[])

{

int rank, size, len;

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank); MPI\_Comm\_size(MPI\_COMM\_WORLD, &size); printf("Hello, world, I am %d of %d\n",rank, size); MPI\_Finalize(); return 0;

}

Compile the program

nllabc2d22@nllabc2d-22:~/opt/openmpi/bin$ mpicc hello.c

Execute the program using 2 cores nllabc2d22@nllabc2d-22:~/opt/openmpi/bin$ mpirun -np 2 ./a.out

# Hello, world, I am 0 of 2 Hello, world, I am 1 of 2

Execute the program using 4 cores

nllabc2d22@nllabc2d-22:~/opt/openmpi/bin$ mpirun -np 4 ./a.out

# Hello, world, I am 0 of 4

# Hello, world, I am 3 of 4

# Hello, world, I am 1 of 4 Hello, world, I am 2 of 4 Program to transfer data from core 0 to core 1.

#include <stdio.h>

#include "mpi.h"

int main(int argc, char\* argv[])

{

int rank, size, len; int num=10;

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank); MPI\_Comm\_size(MPI\_COMM\_WORLD, &size); if(rank == 0)

{

printf("Sending message containing: %d from rank %d\n", num,rank);

MPI\_Send(&num, 1, MPI\_INT, 1, 1, MPI\_COMM\_WORLD);

} else

{

printf(" at rank %d\n",rank);

MPI\_Recv(&num, 1, MPI\_INT, 0, 1, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE); printf("Received message containing: %d at rank %d\n", num,rank);

}

MPI\_Finalize(); return 0;

}

Sending message containing: 10 from rank 0 at rank 1 at rank 3

Received message containing: 10 at rank 1 at rank 2

# /\*\*\*\*\*\* The cores 2 and will be in waiting mode … Press Ctrl+z to end the execution \*\*\*\*\*\*\*/ Assignment program: Add 20 numbers in an array using 4 cores

#include <stdio.h>

#include "mpi.h"

int main(int argc, char\* argv[])

{

int rank, size; int num[20]; //N=20, n=4

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank); MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

for(int i=0;i<20;i++)

num[i]=i+1;

if(rank == 0){ int s[4];

printf("Distribution at rank %d \n", rank);

for(int i=1;i<4;i++)

MPI\_Send(&num[i\*5], 5, MPI\_INT, i, 1, MPI\_COMM\_WORLD); //N/n i.e. 20/4=5 int sum=0, local\_sum=0; for(int i=0;i<5;i++)

{

local\_sum=local\_sum+num[i];

}

for(int i=1;i<4;i++)

{

MPI\_Recv(&s[i], 1, MPI\_INT, i, 1, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

}

printf("local sum at rank %d is %d\n", rank,local\_sum); sum=local\_sum; for(int i=1;i<4;i++)

sum=sum+s[i];

printf("final sum = %d\n\n",sum);

}

else

{

int k[5];

MPI\_Recv(k, 5, MPI\_INT, 0, 1, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE); int local\_sum=0; for(int i=0;i<5;i++)

{

local\_sum=local\_sum+k[i];

}

printf("local sum at rank %d is %d\n", rank, local\_sum);

MPI\_Send(&local\_sum, 1, MPI\_INT, 0, 1, MPI\_COMM\_WORLD);

}

MPI\_Finalize();

return 0;

}

Distribution at rank 0 local sum at rank 1 is 40 local sum at rank 2 is 65 local sum at rank 3 is 90 local sum at rank 0 is 15 final sum = 210

/\*\*\*\*\*\* students can be asked to take dynamic values for N, n and array \*\*\*\*\*\*\*\*\*\*\*\*/

**5.clock sync**

**// Client.cpp**

#include <stdio.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <string.h>

#include <iostream>

#include <stdlib.h> /\* srand, rand \*/

#include <cstdlib>

#include <ctime>

#include <vector>

#define PORT 8080

using namespace std;

// function for string delimiter

vector<string> split(string s, string delimiter) {

size\_t pos\_start = 0, pos\_end, delim\_len = delimiter.length();

string token;

vector<string> res;

while ((pos\_end = s.find (delimiter, pos\_start)) != string::npos) {

token = s.substr (pos\_start, pos\_end - pos\_start);

pos\_start = pos\_end + delim\_len;

res.push\_back (token);

}

res.push\_back (s.substr (pos\_start));

return res;

}

int main(int argc, char const \*argv[])

{

srand((unsigned int)time(NULL)); // avoid always same output of rand()

float client\_local\_clock = rand() % 10; // range from 0 to 9

printf("Client starts. Client pid is %d \n", getpid());

printf("Client local clock is %f \n\n", client\_local\_clock);

int client\_socket\_fd, valread;

char client\_read\_buffer[1024] = {0};

struct sockaddr\_in server\_addr;

server\_addr.sin\_family = AF\_INET;

// server\_addr.sin\_addr.s\_addr = inet\_addr(argv[1]); // hardcode to 127.0.0.1

server\_addr.sin\_port = htons(PORT);

// Creating socket file descriptor (IPv4, TCP, IP)

if ((client\_socket\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) < 0)

{

printf("\n Client: Socket creation error \n");

return -1;

}

// Converting IPv4 and IPv6 addresses from text to binary form,

// from character string src into a network

// address structure in the af address family, then copies the

// network address structure to dst.

if(inet\_pton(AF\_INET, "127.0.0.1", &server\_addr.sin\_addr)<=0)

{

printf("\nClient: Invalid address/ Address not supported \n");

return -1;

}

// Connecting server, return 0 with success, return -1 with error

if (connect(client\_socket\_fd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0)

{

printf("\nClient: Connection Failed \n");

return -1;

}

char server\_ip[INET\_ADDRSTRLEN]="";

inet\_ntop(AF\_INET, &server\_addr.sin\_addr, server\_ip, INET\_ADDRSTRLEN);

printf("Client: connected server(%s:%d). \n", server\_ip, ntohs(server\_addr.sin\_port));

printf("\n\n");

// first round communicattion

// receiving form server

valread = read( client\_socket\_fd , client\_read\_buffer, 1024);

printf("Client: read: '%s'\n",client\_read\_buffer );

// convert char array to string

string recv\_msg = string(client\_read\_buffer);

// reply according to what client receive

if (strcmp(client\_read\_buffer, "Hello from server, please tell me your local clock value.") == 0) {

// prepare msg

string msg\_str = "Hello from client, my local clock value is " + to\_string(client\_local\_clock);

char msg\_char\_array[msg\_str.length() + 1];

strcpy(msg\_char\_array, msg\_str.c\_str());

// sending a message to server

send(client\_socket\_fd , &msg\_char\_array , strlen(msg\_char\_array) , 0 );

printf("Client: sent message: '%s'\n", msg\_char\_array);

}

//

// second round communicattion

//

// receiving form server

valread = read( client\_socket\_fd , client\_read\_buffer, 1024);

printf("Client: read: '%s'\n",client\_read\_buffer );

// convert char array to string

recv\_msg = string(client\_read\_buffer);

if (recv\_msg.find("From server, your clock adjustment offset is") != string::npos){ // if latter is a substring of former

string substr\_after\_lastbutone\_space;

string substr\_after\_last\_space;

vector<string> split\_str = split(recv\_msg, " ");

substr\_after\_lastbutone\_space = split\_str[ split\_str.size() - 2 ];

substr\_after\_last\_space = split\_str[ split\_str.size() - 1 ];

cout << "Client: received local clock adjustment offset (string) is " << substr\_after\_lastbutone\_space << " " << substr\_after\_last\_space << endl;

float substr\_after\_last\_space\_f = stof(substr\_after\_last\_space);

cout << "Client: received local clock adjustment offset (float) is " << substr\_after\_lastbutone\_space << " " << substr\_after\_last\_space\_f << endl;

char oper\_char\_array[substr\_after\_lastbutone\_space.length() + 1];

strcpy(oper\_char\_array, substr\_after\_lastbutone\_space.c\_str());

if (strcmp(oper\_char\_array, "add") == 0 ){

client\_local\_clock += substr\_after\_last\_space\_f;

}else if (strcmp(oper\_char\_array, "minus") == 0 ){

client\_local\_clock -= substr\_after\_last\_space\_f;

}

printf("Client local clock is %f \n\n", client\_local\_clock);

}

close(client\_socket\_fd);

return 0;

}

**//Server.cpp**

#include <iostream>

#include <iomanip>

#include <cstdlib>

#include <unistd.h>

#include <stdio.h>

#include <sys/socket.h>

#include <stdlib.h>

#include <netinet/in.h>

#include <string.h>

#include <arpa/inet.h>

#include <vector>

#include <cstdlib>

#include <ctime>

#define PORT 8080

using namespace std;

// function for string delimiter

vector<string> split(string s, string delimiter) {

size\_t pos\_start = 0, pos\_end, delim\_len = delimiter.length();

string token;

vector<string> res;

while ((pos\_end = s.find (delimiter, pos\_start)) != string::npos) {

token = s.substr (pos\_start, pos\_end - pos\_start);

pos\_start = pos\_end + delim\_len;

res.push\_back (token);

}

res.push\_back (s.substr (pos\_start));

return res;

}

int main(int argc, char \*argv[])

{

// /\* deal with input arguments\*/

// std::cout << "print arguments:\nargc == " << argc << '\n';

// for(int ndx{}; ndx != argc; ++ndx) {

// std::cout << "argv[" << ndx << "] == " << argv[ndx] << '\n';

// }

// std::cout << "argv[" << argc << "] == "

// << static\_cast<void\*>(argv[argc]) << '\n';

srand((unsigned int)time(NULL)); // avoid always same output of rand()

float server\_local\_clock = rand() % 10; // range from 0 to 9

vector<float> clients\_local\_clocks;

printf("Sever starts. Server pid is %d \n", getpid());

printf("Server local clock is %f \n\n", server\_local\_clock);

// Socket Cite: https://www.geeksforgeeks.org/socket-programming-cc/?ref=lbp

int server\_socket\_fd, new\_socket, valread;

vector<int> client\_sockets;

vector<string> client\_ips;

vector<int> client\_ports;

struct sockaddr\_in server\_address;

server\_address.sin\_family = AF\_INET; // IPv4

server\_address.sin\_addr.s\_addr = INADDR\_ANY; // localhost

server\_address.sin\_port = htons( PORT ); // 8080

int opt = 1; // for setsockopt

// Creating socket file descriptor (IPv4, TCP, IP)

if ((server\_socket\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) == 0)

{

perror("Server: socket failed");

exit(EXIT\_FAILURE);

}

// Optional: it helps in reuse of address and port. Prevents error such as: “address already in use”.

if (setsockopt(server\_socket\_fd, SOL\_SOCKET, SO\_REUSEADDR | SO\_REUSEPORT,

&opt, sizeof(opt)))

{

perror("Server: setsockopt");

exit(EXIT\_FAILURE);

}

// Forcefully attaching socket to the port 8080

if (bind(server\_socket\_fd, (struct sockaddr \*)&server\_address,

sizeof(server\_address))<0)

{

perror("Server: bind failed");

exit(EXIT\_FAILURE);

}

// Putting the server socket in a passive mode, waiting for the client to approach the server to make a connection

// The backlog=7, defines the maximum length to which the queue of pending connections for sockfd may grow.

// If a connection request arrives when the queue is full, the client may receive an error with an indication of ECONNREFUSED.

if (listen(server\_socket\_fd, 7) < 0)

{

perror("Server: listen");

exit(EXIT\_FAILURE);

}

printf("Server: server is listening ...\n\nYou can open one or multiple new terminal windows now to run ./client\n");

int clients\_ctr = 0;

// Setting up buffer for receiving msg

char recv\_buf[65536];

memset(recv\_buf, '\0', sizeof(recv\_buf));

int in\_client\_enough = 0;

while ( in\_client\_enough == 0) { // block on accept() until positive fd or error

struct sockaddr\_in client\_addr;

socklen\_t length = sizeof(client\_addr);

// Extracting the first connection request on the queue of pending connections for the listening socket (server\_socket\_fd)

// Creates a new connected socket, and returns a new file descriptor referring to that socket

if ((new\_socket = accept(server\_socket\_fd, (struct sockaddr \*)&client\_addr,

(socklen\_t\*)&length))<0)

{

perror("Server: accept");

exit(EXIT\_FAILURE);

}

clients\_ctr ++;

printf("\nYou have connected %d client(s) now.", clients\_ctr);

// converting the network address structure src in the af address family into a character string.

char client\_ip[INET\_ADDRSTRLEN] = "";

inet\_ntop(AF\_INET, &client\_addr.sin\_addr, client\_ip, INET\_ADDRSTRLEN);

printf("Server: new client accepted. client ip and port: %s:%d\n", client\_ip, ntohs(client\_addr.sin\_port));

// store new client connection into array

client\_sockets.push\_back(new\_socket);

client\_ips.push\_back(client\_ip);

client\_ports.push\_back(ntohs(client\_addr.sin\_port));

printf("current connected clients amount is %d \n", int(client\_sockets.size()) );

cout << "Do you have enought clients? (please input '1' for yes, '0' for no):" ;

cin >> in\_client\_enough;

if (in\_client\_enough == 0){

cout << "OK. Please continute opening one or multiple new terminal windows to run ./client\n" << endl;

}else if (in\_client\_enough != 1){

cout << "Unrecognized input has been considered as 0. You can create one more client.\n" << endl;

in\_client\_enough = 0;

}

}

printf("\nClients creation finished! There are totally %d connected clients.\n", int(client\_sockets.size()) );

printf("Asking all clients to report their local clock value ... \n\n\n");

for (int i = 0; i < client\_sockets.size(); i++){

// sending a message to client

const char \*msg = "Hello from server, please tell me your local clock value.";

send(client\_sockets[i] , msg , strlen(msg) , 0 );

printf("Server: sent to client(%s:%d): '%s'\n", client\_ips[i].c\_str(), client\_ports[i], msg);

// receiving

while(recv(client\_sockets[i], recv\_buf, sizeof(recv\_buf), 0) > 0 ){

printf("Server: recv from client(%s:%d): '%s' \n", client\_ips[i].c\_str(), client\_ports[i], recv\_buf);

// convert char array to string

string recv\_msg = string(recv\_buf);

if (recv\_msg.find("Hello from client, my local clock value is") != string::npos){

string substr\_after\_last\_space;

vector<string> split\_str = split(recv\_msg, " ");

substr\_after\_last\_space = split\_str[ split\_str.size() - 1 ];

cout << "Server: received client local clock (string) is " << substr\_after\_last\_space << endl;

float substr\_after\_last\_space\_f = stof(substr\_after\_last\_space);

cout << "Server: received client local clock (float) is " << substr\_after\_last\_space\_f << endl;

clients\_local\_clocks.push\_back(substr\_after\_last\_space\_f);

}

memset(recv\_buf, '\0', strlen(recv\_buf));

break;

}

}

printf("\n\n");

// average clock values

float all\_clock\_sum = server\_local\_clock;

for (int i = 0; i < clients\_local\_clocks.size(); i++){

all\_clock\_sum += clients\_local\_clocks[i];

}

float avg\_clock = all\_clock\_sum / (client\_sockets.size() + 1);

// tell clients how to adjust

for (int i = 0; i < client\_sockets.size(); i++){

// prepare msg

float offset = clients\_local\_clocks[i] - avg\_clock;

string operation;

if (offset >= 0){

operation = "minus";

}else{

operation = "add";

offset = 0 - offset;

}

string msg\_str = "From server, your clock adjustment offset is " + operation + " " + to\_string(offset);

char msg\_char\_array[msg\_str.length() + 1];

strcpy(msg\_char\_array, msg\_str.c\_str());

// sending a message to client

send(client\_sockets[i] , &msg\_char\_array , strlen(msg\_char\_array) , 0 );

printf("Server: sent to client(%s:%d): '%s'\n", client\_ips[i].c\_str(), client\_ports[i], msg\_char\_array);

}

// adjust self

server\_local\_clock += avg\_clock - server\_local\_clock;

printf("\n\nServer new local clock is %f \n\n", server\_local\_clock);

printf("Server: server stopped. \n");

close(server\_socket\_fd);

return 0;

}

**6.token ring**

**Mutual Exclusion Using Token Ring Algorithm**

**SOURCE CODE:**

**TokenServer.java**

import java.io.\*;  
import java.net.\*;

public class TokenServer  
{  
    public static void main(String agrs[])throws Exception  
        {  
             
            while(true)  
            {  
            Server sr=new Server();  
            sr.recPort(8000);  
            sr.recData();  
            }  
        }  
}

class Server  
{  
     
    boolean hasToken=false;  
    boolean sendData=false;  
    int recport;  
     
    void recPort(int recport)  
    {  
        this.recport=recport;  
    }

    void recData()throws Exception  
    {  
        byte buff[]=new byte[256];  
        DatagramSocket ds;  
        DatagramPacket dp;  
        String str;  
         
        ds=new DatagramSocket(recport);  
        dp=new DatagramPacket(buff,buff.length);  
        ds.receive(dp);  
        ds.close();  
         
        str=new String(dp.getData(),0,dp.getLength());  
        System.out.println(“The message is “+str);  
    }  
}

**TokenClient1.java**

import java.io.\*;  
import java.net.\*;

public class TokenClient1  
{  
    public static void main(String arg[]) throws Exception  
        {  
            InetAddress lclhost;  
            BufferedReader br;  
            String str=””;  
            TokenClient12 tkcl,tkser;  
            boolean hasToken;  
            boolean setSendData;  
             
            while(true)  
            {  
            lclhost=InetAddress.getLocalHost();  
            tkcl = new TokenClient12(lclhost);  
            tkser = new TokenClient12(lclhost);         
            //tkcl.setSendPort(9001);  
            tkcl.setSendPort(9004);  
            tkcl.setRecPort(8002);  
            lclhost=InetAddress.getLocalHost();  
            tkser.setSendPort(9000);

            if(tkcl.hasToken == true)  
            {  
                 
System.out.println(“Do you want to enter the Data –> YES/NO”);  
                br=new BufferedReader(new InputStreamReader(System.in));  
                str=br.readLine();  
                if(str.equalsIgnoreCase(“yes”))  
                {    
                    System.out.println(“ready to send”);  
                    tkser.setSendData = true;  
                    tkser.sendData();  
                    tkser.setSendData = false;  
                }  
                else if(str.equalsIgnoreCase(“no”))  
                {  
                    System.out.println(“i m in else”);  
                    //tkcl.hasToken=false;  
                    tkcl.sendData();  
                    tkcl.recData();  
                System.out.println(“i m leaving else”);  
                }  
            }  
            else  
            {  
            System.out.println(“ENTERING RECEIVING MODE…”);  
                tkcl.recData();  
            }  
    }  
}

}

class TokenClient12  
{  
    InetAddress lclhost;  
    int sendport,recport;  
    boolean hasToken = true;  
    boolean setSendData = false;  
    TokenClient12 tkcl,tkser;  
    TokenClient12(InetAddress lclhost)  
    {  
         
        this.lclhost = lclhost;  
    }  
     
    void setSendPort(int sendport)  
    {  
        this.sendport = sendport;  
    }

    void setRecPort(int recport)     
    {  
        this.recport = recport;  
    }

void sendData() throws Exception  
        {  
        BufferedReader br;  
        String str=”Token”;  
        DatagramSocket ds;  
        DatagramPacket dp;  
         
     
        if(setSendData == true)  
        {  
            System.out.println(“sending “);  
            System.out.println(“Enter the Data”);  
            br=new BufferedReader(new InputStreamReader(System.in));  
            str = “ClientOne…..” + br.readLine();  
            System.out.println(“now sending”);  
                     
        }  
            ds = new DatagramSocket(sendport);  
            dp = new DatagramPacket(str.getBytes(),str.length(),lclhost,sendport-1000);  
            ds.send(dp);  
            ds.close();  
            setSendData = false;  
            hasToken = false;  
    }  
     
    void recData()throws Exception  
    {  
        String msgstr;  
        byte buffer[] = new byte[256];  
        DatagramSocket ds;  
        DatagramPacket dp;

        ds = new DatagramSocket(recport);  
        dp = new DatagramPacket(buffer,buffer.length);  
        ds.receive(dp);  
        ds.close();  
        msgstr = new String(dp.getData(),0,dp.getLength());  
        System.out.println(“The data is “+msgstr);  
         
        if(msgstr.equals(“Token”))  
            {  
                hasToken = true;  
            }  
    }  
     
}  
       

**TokenClient2.java**

import java.io.\*;  
import java.net.\*;

public class TokenClient2  
{  
    static boolean setSendData ;  
    static boolean hasToken ;  
    public static void main(String arg[]) throws Exception  
        {  
            InetAddress lclhost;  
            BufferedReader br;  
            String str1;  
            TokenClient21 tkcl;  
            TokenClient21 ser;  
            while(true)  
            {  
            lclhost=InetAddress.getLocalHost();  
            tkcl = new TokenClient21(lclhost);  
            tkcl.setRecPort(8004);         
            tkcl.setSendPort(9002);  
            lclhost=InetAddress.getLocalHost();  
            ser = new TokenClient21(lclhost);  
            ser.setSendPort(9000);  
            System.out.println(“entering if”);     
            if(hasToken == true)  
            {  
                 
System.out.println(“Do you want to enter the Data –> YES/NO”);  
                br=new BufferedReader(new InputStreamReader(System.in));  
                str1=br.readLine();  
                if(str1.equalsIgnoreCase(“yes”))  
                {     
                    System.out.println(“ignorecase”);  
                    ser.setSendData = true;  
                    ser.sendData();  
                    }  
                else if(str1.equalsIgnoreCase(“no”))  
                {  
                    tkcl.sendData();  
                    hasToken=false;  
                }  
            }  
            else  
            {  
            System.out.println(“entering recieving mode”);     
                tkcl.recData();  
                hasToken=true;  
            }  
        }  
    }  
}  
class TokenClient21  
{  
    InetAddress lclhost;  
    int sendport,recport;  
    boolean setSendData = false;  
    boolean hasToken = false;  
    TokenClient21 tkcl;  
    TokenClient21 ser;

    TokenClient21(InetAddress lclhost)  
    {  
         
        this.lclhost = lclhost;  
    }  
     
    void setSendPort(int sendport)  
    {  
        this.sendport = sendport;  
    }

    void setRecPort(int recport)     
    {  
        this.recport = recport;  
    }

    void sendData() throws Exception  
    {  
        System.out.println(“case”);  
        BufferedReader br;  
        String str=”Token”;  
        DatagramSocket ds;  
        DatagramPacket dp;  
         
        if(setSendData == true)  
        {  
            System.out.println(“Enter the Data”);  
            br=new BufferedReader(new InputStreamReader(System.in));  
            str = “ClientTwo…..” + br.readLine();  
        }  
            ds = new DatagramSocket(sendport);  
            dp = new DatagramPacket(str.getBytes(),str.length(),lclhost,sendport-1000);  
            ds.send(dp);  
            ds.close();  
            System.out.println(“Data Sent”);  
            setSendData = false;  
            hasToken = false;  
         
    }  
   

void recData()throws Exception  
    {  
        String msgstr;  
        byte buffer[] = new byte[256];  
        DatagramSocket ds;  
        DatagramPacket dp;  
ds = new DatagramSocket(recport);  
        //ds = new DatagramSocket(4000);  
        dp = new DatagramPacket(buffer,buffer.length);  
        ds.receive(dp);  
        ds.close();  
msgstr = new String(dp.getData(),0,dp.getLength());  
        System.out.println(“The data is “+msgstr);  
        if(msgstr.equals(“Token”))  
            {  
                hasToken = true;  
            }  
    }  
     
}

**OUTPUT :**

**TokerServer.java**

>java TokenServer  
The message is ClientOne…..hello  
The message is ClientTwo…..hii

**TokelClient1.java**

>java TokenC  
Client1  
Do you want to enter the Data –> YES/NO  
yes  
ready to send  
sending  
Enter the Data  
hello  
now sending  
Do you want to enter the Data –> YES/NO  
no  
i m in else

**TokenClient2.java**

>java TokenClient2  
entering if

entering recieving mode  
The data is Token  
entering if  
Do you want to enter the Data –> YES/NO  
yes  
ignorecase  
case  
Enter the Data  
hii  
Data Sent  
entering if  
Do you want to enter the Data –> YES/NO

**7.election algo**

Bully.java

import java.io.InputStream;

import java.io.PrintStream;

import java.util.Scanner;

public class Bully {

static boolean[] state = new boolean[5];

int coordinator;

public static void up(int up) {

if (state[up - 1]) {

System.out.println("Process " + up + " is already up");

} else {

int i;

Bully.state[up - 1] = true;

System.out.println("Process " + up + " held election");

for (i = up; i < 5; ++i) {

System.out.println("Election message sent from process " + up + " to process " + (i + 1));

}

for (i = up + 1; i <= 5; ++i) {

if (!state[i - 1]) continue;

System.out.println("Alive message send from process " + i + " to process " + up);

break;

}

}

}

public static void down(int down) {

if (!state[down - 1]) {

System.out.println("Process " + down + " is already dowm.");

} else {

Bully.state[down - 1] = false;

}

}

public static void mess(int mess) {

if (state[mess - 1]) {

if (state[4]) {

System.out.println("0K");

} else if (!state[4]) {

int i;

System.out.println("Process " + mess + " election");

for (i = mess; i < 5; ++i) {

System.out.println("Election send from process " + mess + " to process " + (i + 1));

}

for (i = 5; i >= mess; --i) {

if (!state[i - 1]) continue;

System.out.println("Coordinator message send from process " + i + " to all");

break;

}

}

} else {

System.out.println("Process " + mess + " is down");

}

}

public static void main(String[] args) {

int choice;

Scanner sc = new Scanner(System.in);

for (int i = 0; i < 5; ++i) {

Bully.state[i] = true;

}

System.out.println("5 active process are:");

System.out.println("Process up = p1 p2 p3 p4 p5");

System.out.println("Process 5 is coordinator");

do {

System.out.println(".........");

System.out.println("1) Up a process.");

System.out.println("2) Down a process");

System.out.println("3) Send a message");

System.out.println("4) Exit");

choice = sc.nextInt();

switch (choice) {

case 1: {

System.out.println("Bring proces up");

int up = sc.nextInt();

if (up == 5) {

System.out.println("Process 5 is co-ordinator");

Bully.state[4] = true;

break;

}

Bully.up(up);

break;

}

case 2: {

System.out.println("Bring down any process.");

int down = sc.nextInt();

Bully.down(down);

break;

}

case 3: {

System.out.println("Which process will send message");

int mess = sc.nextInt();

Bully.mess(mess);

}

}

} while (choice != 4);

sc.close();

}

}

Ring.java

import java.util.Scanner;

public class Ring {

public static void main(String[] args) {

// TODO Auto-generated method stub

int temp, i, j;

char str[] = new char[10];

Rr proc[] = new Rr[10];

// object initialisation

for (i = 0; i < proc.length; i++)

proc[i] = new Rr();

// scanner used for getting input from console

Scanner in = new Scanner(System.in);

System.out.println("Enter the number of process : ");

int num = in.nextInt();

// getting input from users

for (i = 0; i < num; i++) {

proc[i].index = i;

System.out.println("Enter the id of process : ");

proc[i].id = in.nextInt();

proc[i].state = "active";

proc[i].f = 0;

}

// sorting the processes from on the basis of id

for (i = 0; i < num - 1; i++) {

for (j = 0; j < num - 1; j++) {

if (proc[j].id > proc[j + 1].id) {

temp = proc[j].id;

proc[j].id = proc[j + 1].id;

proc[j + 1].id = temp;

}

}

}

for (i = 0; i < num; i++) {

System.out.print(" [" + i + "]" + " " + proc[i].id);

}

int init;

int ch;

int temp1;

int temp2;

int ch1;

int arr[] = new int[10];

proc[num - 1].state = "inactive";

System.out.println("\n process " + proc[num - 1].id + "select as co-ordinator");

while (true) {

System.out.println("\n 1.election 2.quit ");

ch = in.nextInt();

for (i = 0; i < num; i++) {

proc[i].f = 0;

}

switch (ch) {

case 1:

System.out.println("\n Enter the Process number who initialsied election : ");

init = in.nextInt();

temp2 = init;

temp1 = init + 1;

i = 0;

while (temp2 != temp1) {

if ("active".equals(proc[temp1].state) && proc[temp1].f == 0) {

System.out.println("\nProcess " + proc[init].id + " send message to " + proc[temp1].id);

proc[temp1].f = 1;

init = temp1;

arr[i] = proc[temp1].id;

i++;

}

if (temp1 == num) {

temp1 = 0;

} else {

temp1++;

}

}

System.out.println("\nProcess " + proc[init].id + " send message to " + proc[temp1].id);

arr[i] = proc[temp1].id;

i++;

int max = -1;

// finding maximum for co-ordinator selection

for (j = 0; j < i; j++) {

if (max < arr[j]) {

max = arr[j];

}

}

// co-ordinator is found then printing on console

System.out.println("\n process " + max + "select as co-ordinator");

for (i = 0; i < num; i++) {

if (proc[i].id == max) {

proc[i].state = "inactive";

}

}

break;

case 2:

System.out.println("Program terminated ...");

return ;

default:

System.out.println("\n invalid response \n");

break;

}

}

}

}

class Rr {

public int index; // to store the index of process

public int id; // to store id/name of process

public int f;

String state; // indiactes whether active or inactive state of node

}