PROJECT REPORT SOFTWARE BASED QUEUEING MANAGEMENT SYSTEM

By

Abdul Hadi

Varisha Ajaz

Javeria Irfan

Isra

Navaid Rasheed

Supervisor

Dr. Ayaz ul-Hassan Khan

PAF-KIET
December , 2018

CONTENTS

ABSTRACT

Queues of people form in various situations and locations such as at supermarkets checkouts, banks, airport security, restaurants, etc. A queuing management system is a great need to control queues for better service and customer satisfaction. The process of queue formation and propagation is defined as queuing theory, a mathematical study of waiting queues. In queuing theory, a model is constructed so that queue lengths and waiting time can be predicted. In addition to standard queue processes. Queue system measurements help organizations to enhance its customer service, improve efficiency and reducing costs. Key measurements of typical queuing management systems are: (1) the number of people entering the system, (2) queue length, (3) Average customer wait time, (4) server idle time, (5) total wait time, (6) total system time.

In this project will develop software about queuing management system of restaurant and we take example of Al-Baik restaurant system that will helps

customers to organize their amount of time where they standing in a line, helps the staff to organize their workplace queuing system. We will consider echo of Features in our System, why this queuing system chosen, how to satisfy the single and multiple queuing models and analysis feature for other queuing management software to track the advantages, customer feedback etc.

INTRODUCTION

Queuing is in all aspects in every step in our daily life. It is playing an essential role for business process, reengineering purposes in whole life. It is an essential part of every service or all of the manufacturing process, in the service enterprise customers waiting pending the it's their turn, and in the factory waiting for the raw material pending the their turn. The queues may be real lines at the service center, may be queues at telecom customer service and may be e-mail queues when waiting for a response. So, a priority of the enterprise manager to maintain a short queue and short waiting time, and in order to do so it must be a way to predict the length of the queue and the length of waiting time.

Developing queuing was motivated by its use in the:

- Hospitals / banks / airports / sub-market.
- ATM and other machines.
- Public transport.
- Computer: waiting for a response, so it is useful in information technology to predict the number of requests a computer server will receive.

In computer science, a queue is a particular kind of abstract data type or collection in which the entities in the collection are kept in order and the principal or only operations on the collection are the addition of entities to the rear terminal position, known as enqueuer, and removal of entities from the front terminal position, known as dequeuer. This makes the queue a First in First out (FIFO) data structure.

Aims & Objectives

1- Reduce waiting time of customers.
2- Take orders quickly.
3- Digitization of all transactions.
4- Increase my knowledge of parallel processing and distributed system.

Project Scope

In this project, I will develop queue system for restaurants to collect orders from the customers using many nodes then send it to kitchen node then kitchen node will send the order number to the screen node to call the customers to come and take his order. First, I will create interface of all nodes using JAVA. Second, I will create named Pipe (FIFO) for each node in C, except the kitchen node it should have two (FIFO) one for receive the orders from the order nodes and second to send number of order to the caller node. Lastly, I will create MPI in C to pass orders and management the communication between all components of the restaurant queueing system.

System Architecture

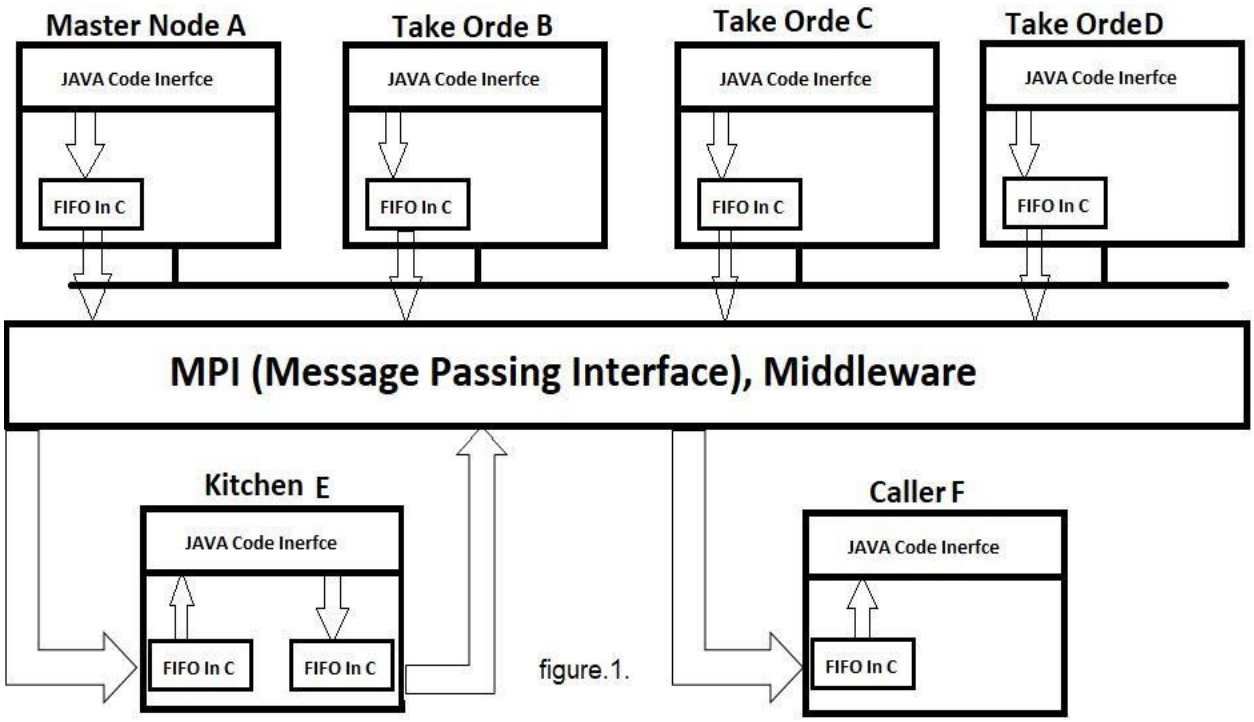
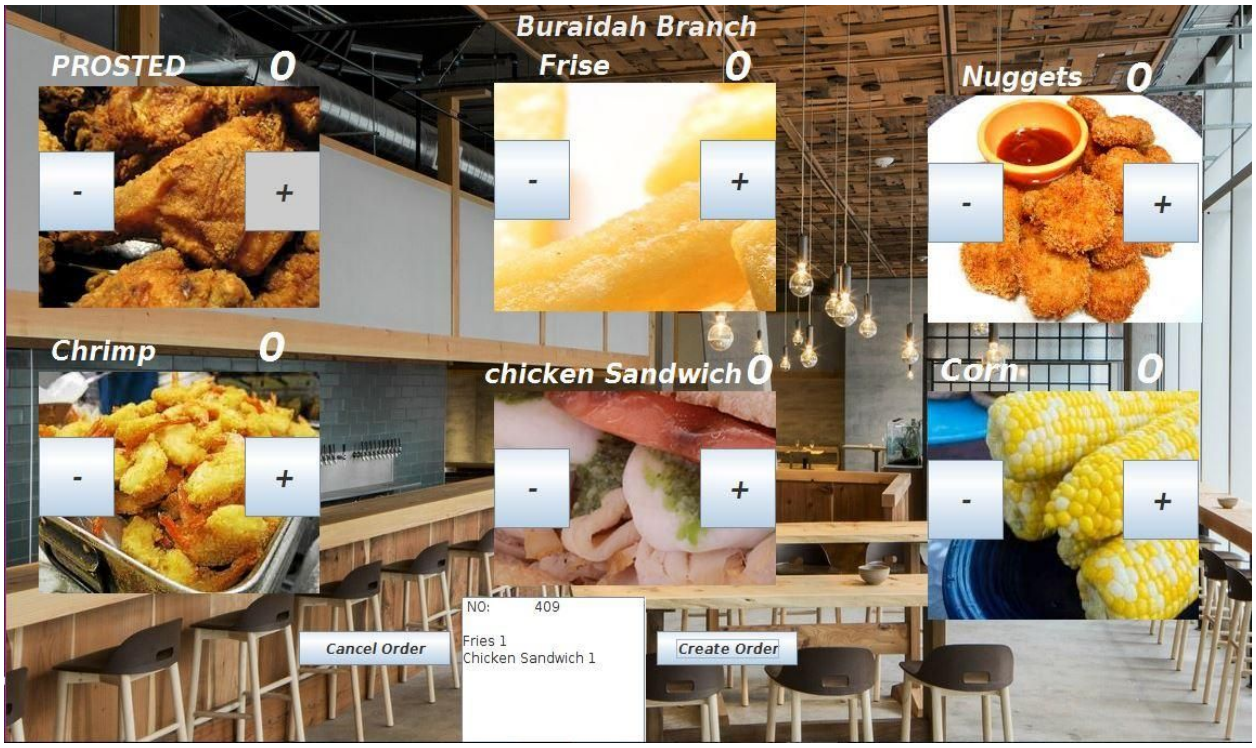


figure.1.

In this architecture showing in figure 1 the Orders of customers will collect by employee from nodes B, C, and D using interface, that have written in JAVA, then send it to FIFOs that is inside each node connect with interfaces from one side and to the MPI (Message Passing Interface) from another side. Then it will pass the orders to the kitchen node (E) through the MPI, the kitchen well receive the orders through the FIFO then send it to the JAVA interface, After prepare the order the Chef will click on the Done button that will appear in front of each order. After the order is done will send it to another FIFO inside same node (E). Then the second FIFO inside the kitchen node (E) will send the order to the MPI. The caller node (F) will receive the order number to call the customers from MPI through the FIFO of node (F).

System Code Samples

First: Orders java interface.



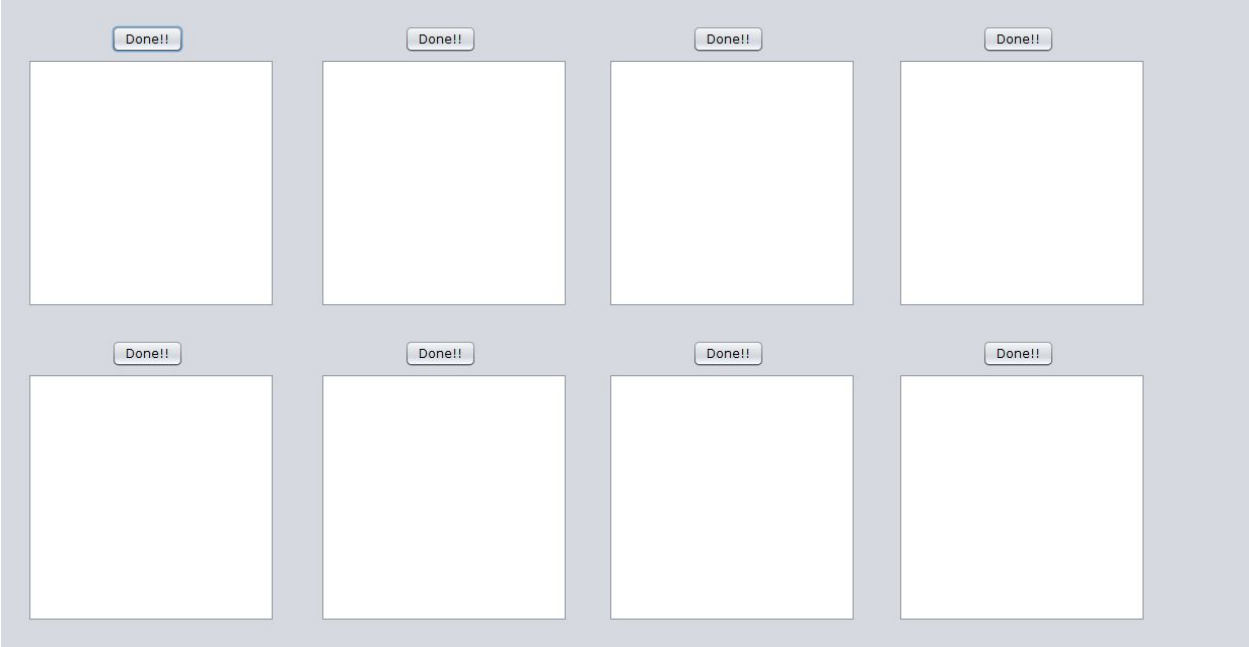
```
14      int numberOfProst ;
15      int numberOfFries ;
16      int numberOfNugget;
17      int numberOfChrimp;
18      int numberOfSandwich;
19      int numberOfCorn;
20
21
22      int NodeNumber ;// number of Machine from 1 to 6 cont .........
23      int InvoiceNumberStart ; // start number of invoice
24      int InvoiceNumberEnd;
25      int numberOf_Meals; // number of meal showin on the screen from 1 to 6 cont ......
26
27      String Meal1;
28      String Meal2;
29      String Meal3;
30      String Meal4;
31      String Meal5;
32      String Meal6;
33      String Restaurant_Name;
34      String  food1="";
35
```

```
36      public void restCounter (){
37      // Counters of The Meals
38          numberOfProst = 0;
39          numberOfFries=0;
40          numberOfNugget=0;
41          numberOfChrimp=0;
42          numberOfSandwich=0;
43          numberOfCorn=0;
44            ProstCounter.setText(Integer.toString(numberOfProst));
45            FriseCounter.setText(Integer.toString(numberOfFries));
46            NuggetsCounter.setText(Integer.toString(numberOfNugget));
47            ChrimpCounter.setText(Integer.toString(numberOfChrimp));
48            SandwichCounter.setText(Integer.toString(numberOfSandwich));
49            CornCounter.setText(Integer.toString(numberOfCorn));
50      }
51
52
53       public TakeOrders() {
54
55          this.numberOfProst = 0;
56          this.numberOfFries=0;
57          this.numberOfNugget=0;
```

```
58          this.numberOfChrimp=0;
59          this.numberOfSandwich=0;
60          this.numberOfCorn=0;
61
62          initComponents();
63          Properties prop = new Properties();
64          InputStream inp ;
65
66          try {
67              inp=new FileInputStream("/home/aamer90/NetBeansProjects/QueueSystem/src/ConfigFiles/Config.txt");
68              prop.load(inp);
69              numberOf_Meals=Integer.parseInt(prop.getProperty("numberOfMeals"));
70
71              NodeNumber = Integer.parseInt(prop.getProperty("Node_Number"));
72              Restaurant_Name=prop.getProperty("RestaurantName");
73              Meal1=prop.getProperty("Meal_1");
74              Meal2=prop.getProperty("Meal_2");
75              Meal3=prop.getProperty("Meal_3");
76              Meal4=prop.getProperty("Meal_4");
77              Meal5=prop.getProperty("Meal_5");
78              Meal6=prop.getProperty("Meal_6");
79
```

```
752
            food1+=" NO:       " +InvoiceNumberStart +"\n\n";
754         String Meal_1=Meal1+" "+numberOfProst+"\n";
755         String Meal_2=Meal2+" "+numberOfFries+"\n";
756         String Meal_3=Meal3+" "+numberOfNugget+"\n";
757         String Meal_4=Meal4+" "+numberOfChrimp+"\n";
758         String Meal_5=Meal5+" "+numberOfSandwich+"\n";
759         String Meal_6=Meal6+" "+numberOfCorn+"\n";
760
761          if(numberOfProst>0){
762          food1+= Meal_1;
763          }
764
765          if(numberOfFries>0){
766          food1+= Meal_2;
767          }
768
769          if(numberOfNugget>0){
770          food1+= Meal_3;
771          }
772
```

Second: Kitchen java interface:




```
1   package kitchen;
2
3   import java.awt.Button;
    import java.awt.Color;
    import java.awt.Font;
    import java.awt.event.ActionEvent;
    import java.awt.event.ActionListener;
8   import java.io.FileNotFoundException;
9   import java.io.IOException;
10  import java.io.RandomAccessFile;
11  import java.util.ArrayList;
12  import java.util.logging.Level;
13  import java.util.logging.Logger;
14  import javax.swing.JTextArea;
15
16  public class NewJFrame extends javax.swing.JFrame {
17
18      ArrayList<String> list, delList;
19      ArrayList<JTextArea> listOfTextbox;
20
21      int WIDTH_OF_TEXT = 600, HEIGHT_OF_TEXT = 100;
22      int GET_WIDTH_OF_SCREEN = 0, GET_HEIGHT_OF_SCREEN = 0;
23      /**
24       * Creates new form NewJFrame
25       */
```

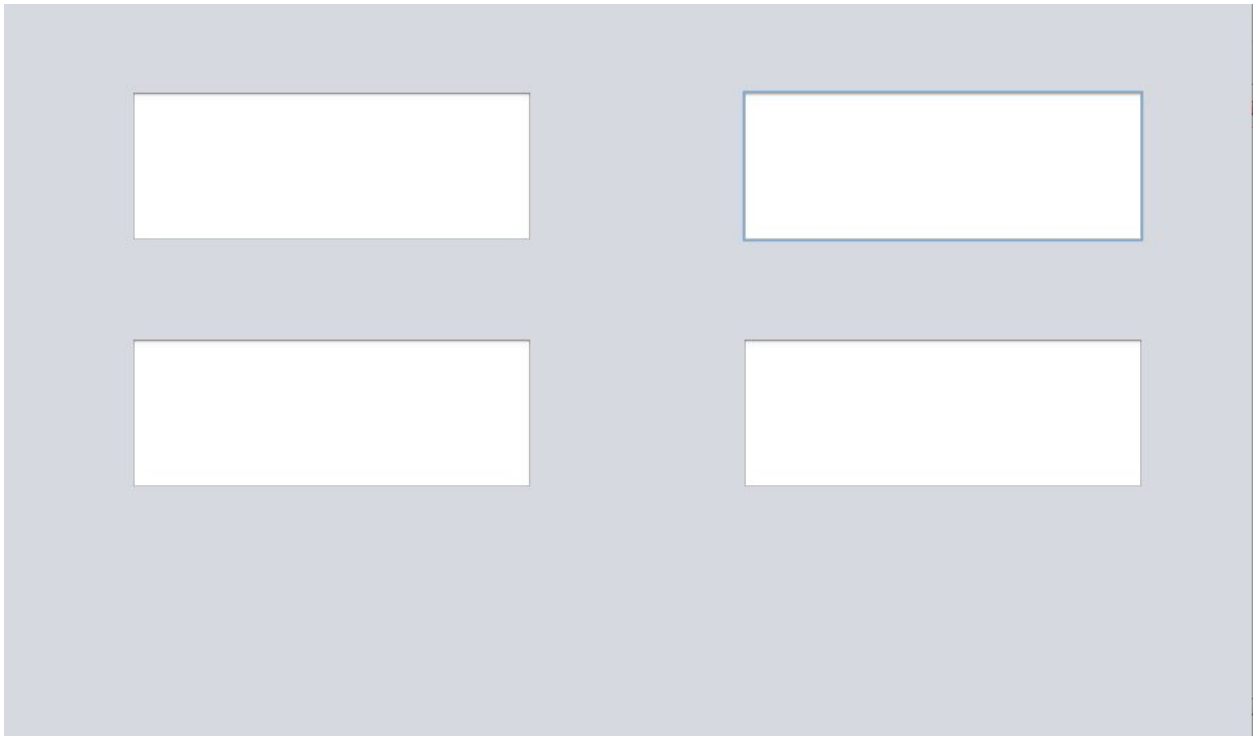
```
25          */
26         public NewJFrame() {
27             initComponents();
28             list = new ArrayList<>();
29             delList = new ArrayList<>();
30 
31             listOfTextbox = new ArrayList<>();
32             listOfTextbox.add(txtbox1);
33             listOfTextbox.add(txtbox2);
34             listOfTextbox.add(txtbox3);
35             listOfTextbox.add(txtbox4);
36             listOfTextbox.add(txtbox5);
37             listOfTextbox.add(txtbox6);
38             listOfTextbox.add(txtbox7);
39             listOfTextbox.add(txtbox8);
40 
41             for (int i = 0; i < listOfTextbox.size(); i++) {
42                 listOfTextbox.get(i).setText("");
43                 listOfTextbox.get(i).setEditable(false);
44 
45             }
46             // Connect to the named pipe+
47 
48             Thread thred = new Thread(){
49                 public void run(){
```

```
50         while(true){
51 
52            try {
53 
54                // Write request to the pipe
55                RandomAccessFile pipe = new RandomAccessFile("/home/aamer90/NetBeansProject
56                        + "/QueueSystem/writefifo", "rw");
57 
58                String order="";
59 
60                try {
61                    for(;pipe.readBoolean();){
62                        order += pipe.readLine()+"\n";//5
63                    }
64 
65                    System.out.println(order);
66                    list.add(order);
67                    setMessage();
68                    // Close the pipe
69                    pipe.close();
70 
71                } catch (IOException ex) {
72                    System.out.println("ERROR");
73 
74                }
```

```
75
76
77                      } catch (FileNotFoundException ex) {
78                          Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
79                      }
30
31                  }
32                  }
33
34              };
                thred.start();
36
37      }
38
39
90      void labAndbtnRemove(Button btn, JTextArea lab){
91          boxView.remove(btn);
92          boxView.remove(lab);
93      }
94
```

```
94
95      void setMessage(){
96           for (int i = 0; i < list.size(); i++) {
97              for (int j = 0; j < listOfTextbox.size(); j++) {
98                  if(listOfTextbox.get(j).getText().equals("")){
99                      listOfTextbox.get(j).setText(list.get(i));
100                     delList.add(list.get(i));
101                     break;
102                 }
103             }
104         }
105          for (int i = 0; i < delList.size(); i++) {
106             list.remove(delList.get(i));
107         }
108     }
109
110     /**
111      * This method is called from within the constructor to initialize the form.
112      * WARNING: Do NOT modify this code. The content of this method is always
113      * regenerated by the Form Editor.
```

Third: Caller java interface:



```
1   package caller;
2
3   import java.io.FileNotFoundException;
4   import java.io.IOException;
5   import java.io.RandomAccessFile;
6   import java.util.ArrayList;
7   import java.util.logging.Level;
8   import java.util.logging.Logger;
9   import javax.swing.JTextArea;
10
11  public class NewJFrame extends javax.swing.JFrame
12
13      ArrayList<String> list, delList;
14      ArrayList<JTextArea> listOfTextbox;
15
16      public NewJFrame() {
17          initComponents();
18          list = new ArrayList<>();
19          delList = new ArrayList<>();
20          /*just for test the code
21          list.add("120");
22          list.add("222");
23          list.add("333");
24          list.add("444");
25          list.add("555");
```

```
26          list.add("666");
27          list.add("777");
28          */
29          listOfTextbox = new ArrayList<>();
30          listOfTextbox.add(txtbox2);
31          listOfTextbox.add(txtbox1);
32          listOfTextbox.add(txtbox3);
33          listOfTextbox.add(txtbox4);
34
35           for (int i = 0; i < listOfTextbox.size(); i++) {
36              listOfTextbox.get(i).setText("");
37          }
38           for (int i = 0; i < listOfTextbox.size(); i++) {
39              listOfTextbox.get(i).setEditable(false);
40          }
41
42           Thread thred = new Thread(){
43              public void run(){
44                   while(true){
45
46                      try {
47
48                          // Write request to the pipe
49
50                          RandomAccessFile pipe = new RandomAccessFile("/home/aamer90/NetBeansProject
```

```
52                          // prop.load(pipe);
53                          String order="";
54
55                          try {
56                              for(;pipe.readBoolean();){
57                                  order += pipe.readLine()+"\n";
58                              }
59
60                              System.out.println(order);
61                              list.add(order);
62                              setMessage();
63                              // Close the pipe
64                              pipe.close();
65
66                          } catch (IOException ex) {
67                              System.out.println("ERROR");
68
69                          }
70
71                      } catch (FileNotFoundException ex) {
72                          System.out.println("ERROR");
73                      }} }
74           };
75          thred.start();
76      }
```

```
77
78      void setMessage(){
79          for (int i = 0; i < list.size(); i++) {
80             for (int j = 0; j < listOfTextbox.size(); j++) {
81                 if(listOfTextbox.get(j).getText().equals("")){
82                     listOfTextbox.get(j).setText(list.get(i));
83                     delList.add(list.get(i));
84
85                     break;
86                 }
87             }
88          }
89          for (int i = 0; i < delList.size(); i++) {
90             list.remove(delList.get(i));
91          }
92          for(int j = 0; j < listOfTextbox.size(); j++)
93              delText(j);
94
95      }
96
97      void delText(int index){
98          if(!listOfTextbox.get(index).getText().equals("")){
99              final int getIndex = index;
100         new Thread(){
101
```

```
101
102             @Override
                public void run() {
                    super.run(); //To change body of generated methods, choose Tools | Templates.
105                 try {
106                     sleep(20000);
107                     listOfTextbox.get(getIndex).setText("");
108                     sleep(getIndex * 1000);
109                     setMessage();
110                 } catch (InterruptedException ex) {
111                     Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
112                 }
113             }
114
115         }.start();
116         }
117     }
118
```

Forth: Named Pipe (FIFO) Code written in C.

```c
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <unistd.h>
#include <mpi.h>
#include <string.h>
#include <pthread.h>
#include <omp.h>
#define MAX_BUF 1024

//void *functionOfThread(void * parm);

int main(int argv, char *argc[])
{
    int myrank, process, err;
    int fd,fd1,fd2,fd3;
    char * myfifo = "/home/aamer90/NetBeansProjects/QueueSystem/FifoFile";
    char * myfifo1 = "/home/aamer90/NetBeansProjects/QueueSystem/writefifo";
    char * myfifo2 = "/home/aamer90/NetBeansProjects/QueueSystem/rwfifio";
    char * myfifo3 = "/home/aamer90/NetBeansProjects/QueueSystem/callfifio";
    mkfifo(myfifo, 0666);
    mkfifo(myfifo1, 0666);
    mkfifo(myfifo2, 0666);
    mkfifo(myfifo3, 0666);

    char buf[MAX_BUF];

    err = MPI_Init(&argv, &argc);



    /* open, read, and display the message from the FIFO */
```

```
while(1)
{

    int length;
    int rc;
    err = MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
    err = MPI_Comm_size(MPI_COMM_WORLD, &process);


    if(myrank==0)
    {

        fd = open(myfifo, O_RDONLY);
        buf[0]=0;
        if((rc = read(fd, buf, MAX_BUF)) > 0)//step2
        {
            buf[rc] = 0;
            printf("Receivedp0: %s\n", buf);
            length= strlen(buf)+1;
            MPI_Send(&length,1, MPI_INT, 1, 1, MPI_COMM_WORLD);//3
            MPI_Send(buf,length,MPI_CHAR,1,0,MPI_COMM_WORLD);
            printf("SIZE: %d \n\n",length);
        }
        close(fd);
    }
    else if(myrank==1)
    {

        #pragma omp parallel num_threads(2)
        {

            //char receve[length];
```

```
            int x = omp_get_thread_num();

if(x == 0){
                MPI_Status status;
                MPI_Recv(&length, 1, MPI_INT, 0, 1, MPI_COMM_WORLD, &status);
                MPI_Recv(buf,length,MPI_CHAR,0,0,MPI_COMM_WORLD,&status);

                buf[length]=0;
                printf("P1rec buf = %s \n\n",buf);
                fd1 = open(myfifo1, O_WRONLY);//4
                write(fd1, buf, length);
                close(fd1);
      }


            /*
                        pthread_t tid;
                        pthread_create(&tid,NULL,functionOfThread,NULL);
            */

else{
            fd2 = open(myfifo2, O_RDONLY);
            buf[0]=0;

            if((rc = read(fd2, buf, MAX_BUF)) > 0)
            {
                buf[rc] = 0;
                printf("Receivedp1 sevd: %s\n", buf);
                int length= strlen(buf)+1;
                MPI_Send(&length,1, MPI_INT, 2, 2, MPI_COMM_WORLD);
                MPI_Send(buf,length,MPI_CHAR,2,3,MPI_COMM_WORLD);
                printf("SIZE: %d \n\n",length);
            }
            close(fd2);
```

```
}
            }// paraalesm
        }

    else
    {


        MPI_Status status;
        MPI_Recv(&length, 1, MPI_INT, 1, 2, MPI_COMM_WORLD, &status);
        MPI_Recv(buf,length,MPI_CHAR,1,3,MPI_COMM_WORLD,&status);

        buf[length]=0;

        fd3 = open(myfifo3, O_WRONLY);
        write(fd3, buf, length);
        printf("Receivedp3: %s\n", buf);
        close(fd3);

    }

}//endOfOuterLoop
err =MPI_Finalize();
return 0;
}
```

To Run it On Different Machines we need to setup a cluster :

Running an MPI Cluster within a LAN

Pre-requisite

Step 1: Configure your `hosts` file

You are gonna need to communicate between the computers and you don’t want to type in the IP addresses every so often. Instead, you can give a name to the various nodes in the network that you wish to

communicate with. `hosts` file is used by your device operating system to map hostnames to IP addresses.

```
$ cat /etc/hosts

127.0.0.1       localhost
172.50.88.34    client
```

```
172.50.88.34    client
```

The `client` here is the machine you'd like to do your computation with. Likewise, do the same about `master` in the client.

Step 2: Create a new user

Though you can operate your cluster with your existing user account, I'd recommend you to create a new one to keep our configurations simple. Let us create a new user `mpiuser`. Create new user accounts with the same username in all the machines to keep things simple.

```
$ sudo adduser mpiuser
```

Follow prompts and you will be good. Please don't use `useradd` command to create a new user as that doesn't create a separate home for new users.

Step 3: Setting up SSH

Your machines are gonna be talking over the network via SSH and share data via NFS, about which we'll talk a little later.

```
$ sudo apt-get install openssh-server
```

And right after that, login with your newly created account

```
$ su - mpiuser
```

Since the `ssh` server is already installed, you must be able to login to other machines by `ssh username@hostname`, at which you will be prompted to enter the password of the `username`. To enable more easier login, we generate keys and copy them to other machines' list of `authorized_keys`.

```
$ ssh-keygen -t dsa
```

You can as well generate RSA keys. But again, it is totally up to you. If you want more security, go with RSA. Else, DSA should do just fine. Now, add the generated key to each of the other computers. In our case, the client machine.

```
$ ssh-copy-id client #ip-address may also be used
```

Do the above step for each of the client machines and your own user (localhost).

This will setup `openssh-server` for you to securely communicate with the client machines. `ssh` all machines once, so they get added to your list of `known_hosts`. This is a very simple but essential step failing which passwordless `ssh` will be a trouble.

Now, to enable passwordless ssh,

```
$ eval `ssh-agent`
$ ssh-add ~/.ssh/id_dsa
```

Now, assuming you've properly added your keys to other machines, you must be able to login to other machines without any password prompt.

```
$ ssh client
```

Note - Since I've assumed that you've created `mpiuser` as the common user account in all of the client machines, this should just work fine. If you've created user accounts with different names in master and client machines, you'll need to work around that.

Step 4: Setting up NFS

You share a directory via NFS in **master** which the **client** mounts to exchange data.

NFS-Server

Install the required packages by

```
$ sudo apt-get install nfs-kernel-server
```

Now, (assuming you are still logged into `mpiuser`), let's create a folder by the name `cloud` that we will share across in the network.

```
$ mkdir cloud
```

To export the `cloud` directory, you create an entry in `/etc/exports`

```
$ cat /etc/exports
/home/mpiuser/cloud *(rw,sync,no_root_squash,no_subtree_check)
```

Here, instead of `*` you can specifically give out the IP address to which you want to share this folder to. But, this will just make our job easier.

After you have made the entry, run the following.

```
$ exportfs -a
```

Run the above command, every time you make a change to `/etc/exports`.

If required, restart the `nfs` server

```
$ sudo service nfs-kernel-server restart
```

NFS-Client

Install the required packages

```
$ sudo apt-get install nfs-common
```

Create a directory in the client's machine with the samename `cloud`

```
$ mkdir cloud
```

And now, mount the shared directory like

```
$ sudo mount -t nfs master:/home/mpiuser/cloud ~/cloud
```

To check the mounted directories,

```
$ df -h
Filesystem                 Size  Used Avail Use% Mounted on
master:/home/mpiuser/cloud  49G   15G   32G  32%
/home/mpiuser/cloud
```

To make the mount permanent so you don't have to manually mount the shared directory everytime you do a system reboot, you can create an entry in your file systems table - i.e., `/etc/fstab` file like this:

```
$ cat /etc/fstab
#MPI CLUSTER SETUP
master:/home/mpiuser/cloud /home/mpiuser/cloud nfs
```

Then you have to change the path in the java applications and in the ReadWrite.c file to the shared location we just created

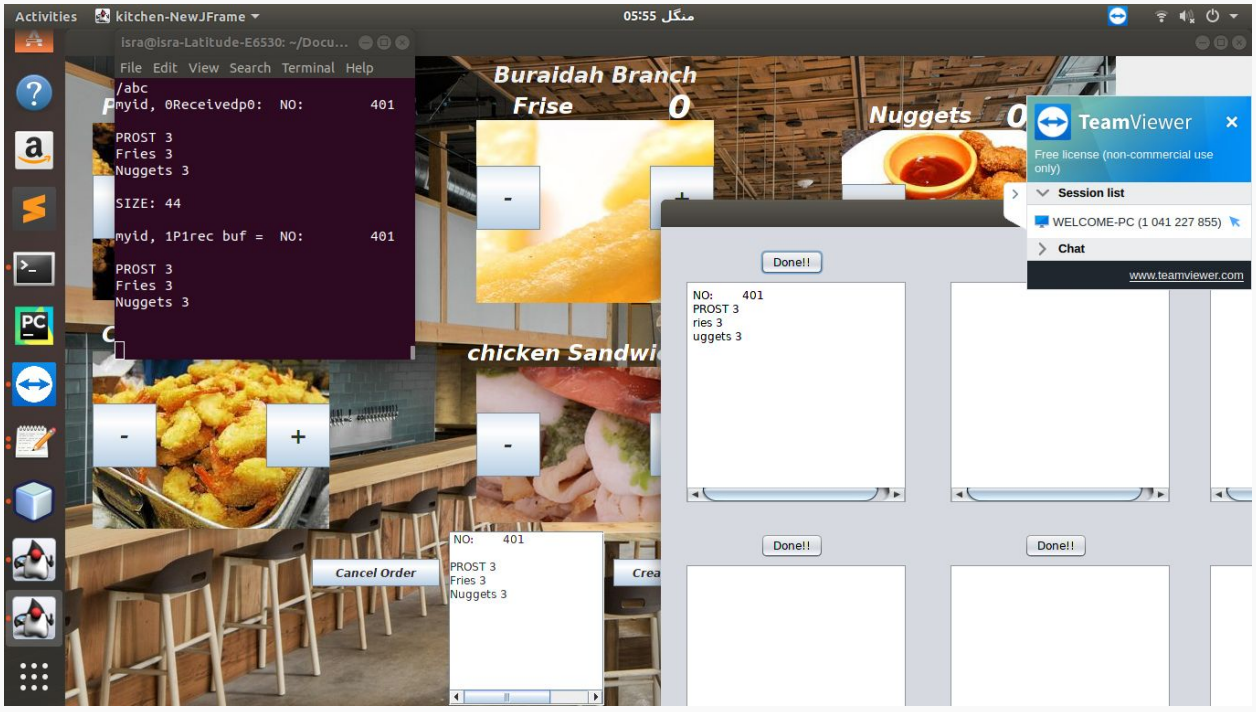
this location

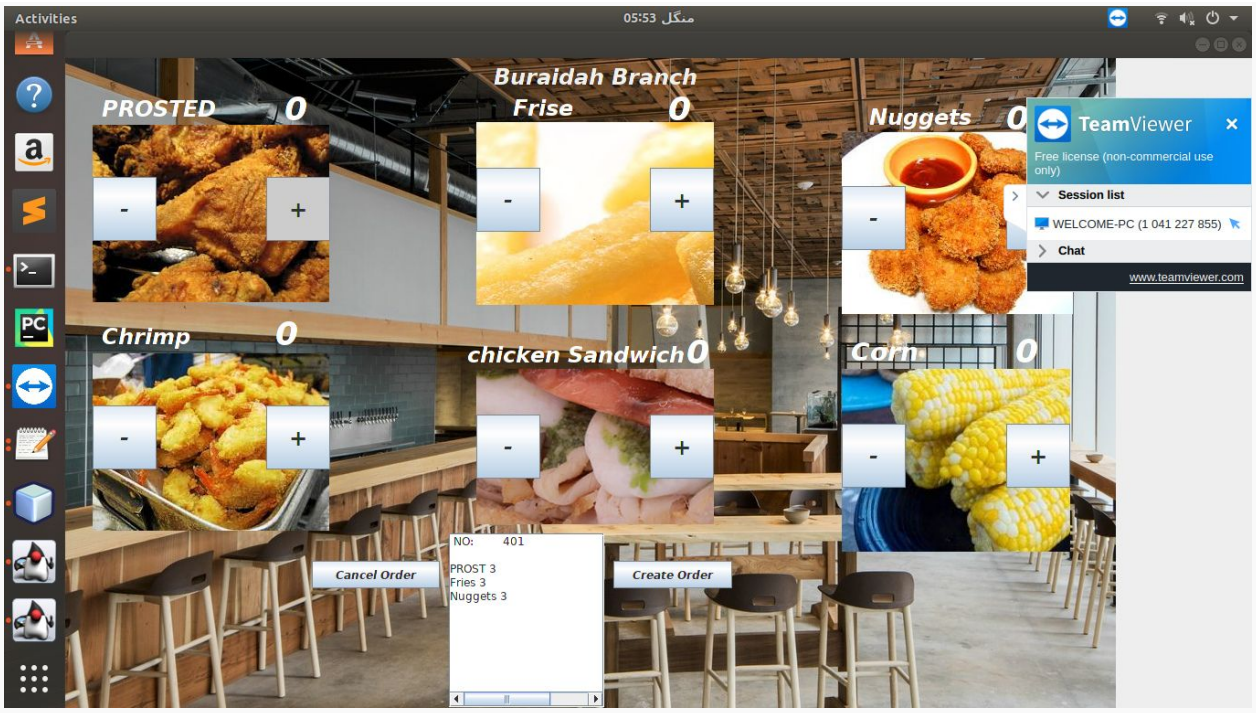
```
$ cd cloud/
$ pwd
/home/mpiuser/cloud
```

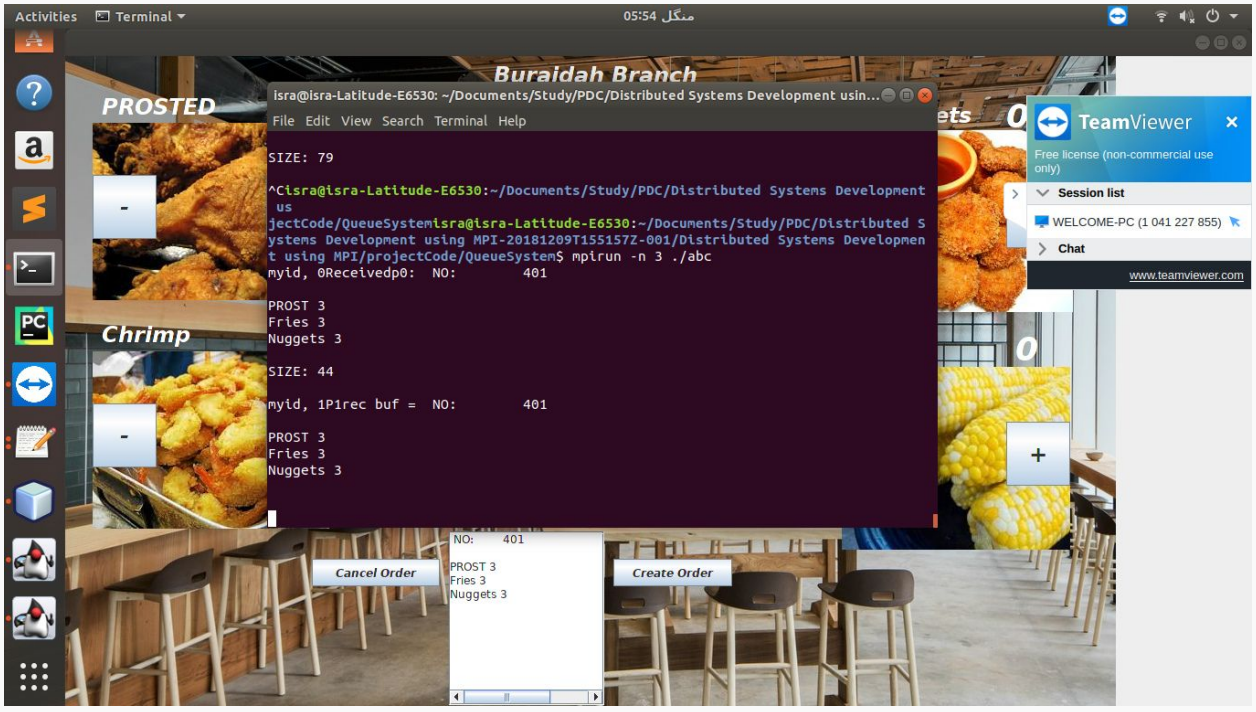
then run the above program by giving the host file

```
$ mpirun -np 3 --hostfile mpi_file ./cpi -fopenmp
```

```
some snaps :
```


Activities
kitchen-NewJFrame
05:55 منگل
isra@isra-Latitude-E6530: ~/Docu...
File Edit View Search Terminal Help
/abc
myid, 0Receivedp0: NO: 401
PROST 3
Fries 3
Nuggets 3
SIZE: 44
myid, 1P1rec buf = NO: 401
PROST 3
Fries 3
Nuggets 3
Buraidah Branch
Frise 0
Nuggets 0
chicken Sandwi
Cancel Order
NO: 401
PROST 3
Fries 3
Nuggets 3
Done!!
NO: 401
PROST 3
ries 3
uggets 3
Done!!
Done!!
TeamViewer
Free license (non-commercial use only)
Session list
WELCOME-PC (1 041 227 855)
Chat
www.teamviewer.com


Activities
05:53 منگل
PROSTED 0
Buraidah Branch
Frise 0
Nuggets 0
Chrimp 0
chicken Sandwich 0
Corn 0
Cancel Order
Create Order
NO: 401
PROST 3
Fries 3
Nuggets 3
TeamViewer
Free license (non-commercial use only)
Session list
WELCOME-PC (1 041 227 855)
Chat
www.teamviewer.com



Link to Code :

https://drive.google.com/open?id=1-Hcls48eOS8HPIk_B34Uh-JdXgKNfW8g