MUTUAL EXCLUSION USING RICART – AGRAWALA ALGORITHM FOR A SALES MANAGEMENT SYSTEM

A PROJECT REPORT

for

ITE3004 – DISTRIBUTED SYSTEMS

by

17BIT0001 - VARUN MUPPALLA

17BIT0119 – ANKUSH KUMAR

17BIT0346 – DINAV PANDIA

SUBMITTED TO:

PROF. DR. SIVAKUMAR N

ABSTRACT

Ricart–Agrawala algorithm is an algorithm to for mutual exclusion in a distributed system proposed by Glenn Ricart and Ashok Agrawala. This algorithm is an extension and optimization of Lamport's Distributed Mutual Exclusion Algorithm. Like Lamport's Algorithm, it also follows permission based approach to ensure mutual exclusion .In this algorithm, two types of messages (request and reply) are used and communication channels are assumed to follow. A site sends a message to all other site to get their permission to enter critical section. A timestamp is given to each critical section request using Lamport's logical clock. Timestamp is used to determine priority of critical section requests. Smaller timestamp gets high priority over larger timestamp. The execution of critical section request is always in the order of their timestamp. We have used this algorithm to apply it to an inventory management system. The applications of the Ricart – Agrawala algorithm are very wide spread and the algorithm can be used in many distributed mutual exclusion environments like banking, airline management and other things.

KEYWORDS – Distributed Systems, Mutual Exclusion, Lamport's Algorithm, Ricart – Agrawala Algorithm

I. INTRODUCTION

The Ricart-Agrawala Algorithm is an algorithm for mutual exclusion on a distributed system. This algorithm is an extension and optimization of Lamport's Distributed Mutual Exclusion Algorithm, by removing the need for display style and display exchange messages. It was developed by Glenn Ricart and Ashok Agrawala. The Ricart-Agrawala Algorithm is definitely an algorithm with regard to mutual exclusion on a distributed system. This algorithm is an extension as well as optimization associated with Lamport's Distributed Mutual Exclusion Algorithm, by removing the need for release messages. It had been developed by Glenn Ricart and Ashok Agrawala.

In our system, we have used the mutual exclusion algorithm to implement a sales management system. The system prevents write operations both at the same time from two different clients. When a client makes an account and tries to make a transaction through it, it allows the client to complete the transaction. When a different transaction tries to occur from another client, the transaction does not pass till the first transaction is completed. As soon as the first transaction is completed, the algorithm moves on to the next transaction in the queue.

However, when the client tries to make a read transaction from the system and there is a write operation being performed in the analogous "critical section", the system allows the user to fetch the data and replies with the data that was performed in the last transaction that had been sealed after writing the data.

II. BACKROUND

MUTUAL EXCLUSION IN DISTRIBUTED SYSTEMS:

Mutual exclusion is a concurrency control property which is introduced to prevent race conditions. It is the requirement that a process cannot enter its critical section while another concurrent process is currently present or executing in its critical section i.e., only one process is allowed to execute the critical section at any given instance of time.

Mutual Exclusion in a Distributed Environment v/s a Distributed Environment:

In single computer system, memory and other resources are shared between different processes. The status of shared resources and the status of users is easily available in the shared memory so with the help of shared variable (For example: Semaphores) mutual exclusion problem can be easily solved.

In Distributed systems, we neither have shared memory nor a common physical clock and there for we cannot solve mutual exclusion problem using shared variables. To eliminate the mutual exclusion problem in distributed system approach based on message passing is used. A site in distributed system does not have complete information of state of the system due to lack of shared memory and a common physical clock.

Requirements of Mutual exclusion Algorithm:

- Avoid Deadlock: Clients should not wait for a transaction that they want to make for an infinitely long time.
- No Starvation: Every site who wants to execute critical section should get an opportunity to execute it in finite time. Any site should not wait indefinitely to execute critical section while other site are repeatedly executing critical section
- Fairness: Each site should get a fair chance to execute critical section. Any request to execute critical section must be executed in the order they are made i.e., Critical section execution requests should be executed in the order of their arrival in the system.
- Fault Tolerance: In case of failure, it should be able to recognize it by itself in order to continue functioning without any disruption.

Solution to distributed mutual exclusion:

As we know shared variables or a local kernel cannot be used to implement mutual exclusion in distributed systems. Message passing is a way to implement mutual exclusion. Below are the three approaches based on message passing to implement mutual exclusion in distributed systems:

Token Based Algorithm:

- A unique token is shared among all the sites.

- If a site possesses the unique token, it is allowed to enter its critical section
- This approach uses sequence number to order requests for the critical section.
- Each requests for critical section contains a sequence number. This sequence number is used to distinguish old and current requests.
- This approach insures Mutual exclusion as the token is unique.

Non-token based approach:

- A site communicates with other sites in order to determine which sites should execute critical section next. This requires exchange of two or more successive round of messages among sites.
- This approach use timestamps instead of sequence number to order requests for the critical section.
- Whenever a site make request for critical section, it gets a timestamp. Timestamp is also used to resolve any conflict between critical section requests.
- All algorithm which follows non-token based approach maintains a logical clock. Logical clocks get updated according to Lamport's scheme

Quorum based approach:

- Instead of requesting permission to execute the critical section from all other sites, Each site requests only a subset of sites which is called a quorum.
- Any two subsets of sites or Quorum contain a common site.
- This common site is responsible to ensure mutual exclusion.

A type of token based algorithm is the Lamport's Algorithm and an extension of this is the Ricart-Agrawala Algorithm.

Lamport's Distributed Mutual Exclusion Algorithm is a permission based algorithm proposed by Lamport as an illustration of his synchronization scheme for distributed systems.
In permission based timestamp is used to order critical section requests and to resolve any conflict between requests.

In Lamport's Algorithm critical section requests are executed in the increasing order of timestamps i.e., a request with smaller timestamp will be given permission to execute critical section first than a request with larger timestamp.

How the Lamport Algorithm was improved to the Ricart-Agrawala Algorithm:

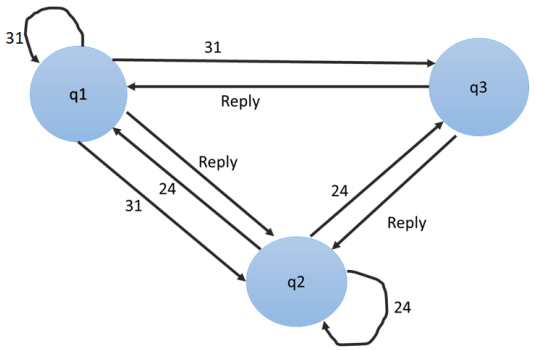
The drawbacks of Lamport's Algorithm are:

- Unreliable approach: failure of any one of the processes will halt the progress of entire system.
- High message complexity: Algorithm requires 3(N-1) messages per critical section invocation.

The Ricart-Agrawala algorithm presumes the communication channels tend to be FIFO. The actual algorithm utilizes two kinds of messages: REQUEST as well as REPLY.A process sends a REQUEST message to any or all additional procedures in order to request their permission to enter the critical section. A process sends a REPLY message to a process to give its permission to that process. Processes use Lamport-style logical clocks to assign a timestamp to critical section requests as well as timestamps are used to decide the priority of requests. Each process pi keeps the actual Request-Deferred array, RDi, the size of which is the same as the number of processes in the system. At first, ∀ i ∀ j : RDi [j]=0. Whenever pi defer the request sent by pj, it sets RDi [j] =1 as well as following it's sent a REPLY message to pj, this sets RDi [j] = 0.

Algorithm:

1) To enter Critical section:
   i. When a site Si wants to enter the critical section, it send a timestamped REQUEST message to all other sites.
   ii. When a site Sj receives a REQUEST message from site Si, It sends a REPLY message to site Si if and only if
   iii. Site Sj is neither requesting nor currently executing the critical section.
   iv. In case Site Sj is requesting, the timestamp of Site Si‘s request is smaller than its own request.
   v. Otherwise the request is deferred by site Sj.
2) To execute the critical section:
   i. Site Si enters the critical section if it has received the REPLY message from all other sites.
   ii. To release the critical section:
3) Upon exiting site Si sends REPLY message to all the deferred requests.



III. CODE / IMPLEMENTATION

Server side:

```
import socket
import time
import threading
local_time=[]
names=[]
account_no=[]
#account_no.append(12)
balance=[]
#balance.append(2000)


ip_addresses=[]
action=[]
flags=[]
choices=[]
req_timestamp=[]
def is_free(account_no_):
 if flags[account_no.index(str(account_no_))]==1:
 return False
 return True
def broad_cast_for_permission(j,requester_time_stamp,acc_no):
 ## acting as client
 print(str(requester_time_stamp))
 print(str(account_no[0]))
 try:
 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM);
print("Broadcasting.....")
 except socket.error as err:
 print("Could not setup socket");
port = 12399;
 print("Ips "+str(ip_addresses))
 ip = ip_addresses[j];
 with s:
 s.connect((ip,port));


 rec = s.recv(1024);
 print(rec.decode());


  data_sent = (str(acc_no)).encode(); ##sending acc no
```

```
s.sendall(data_sent);
 data_sent =(str(requester_time_stamp)).encode();
 s.sendall(data_sent);
rec = s.recv(1024);
 message=rec.decode();
 print(message)
 time.sleep(0.5)
 s.close();
 return message
def richarts_algo():
while True:
 num=len(ip_addresses)
 for i in range(num):
 no_of_permission=0
 acc_no=account_no[0]
print(req_timestamp)
 print("index"+str(i))
 requester_time_stamp=int(req_timestamp[i])
 print("req time "+str(requester_time_stamp))


 for j in range(num):
 if i!=j:
no_of_permission+=int(broad_cast_for_permission(j,requester_time_stamp
,acc_no)) # j is the person to connect with account no is the account
requested for
 if no_of_permission==num-1:
print(broad_cast_for_permission(j,requester_time_stamp,-1))
 time.sleep(2)
 reply_client(ip_addresses[i],choices[i])
def serving():


 try:
 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM);

except socket.error as err:
 print("Socket creation failed");
port = 12365;

 s.bind(('', port));
s.listen();
print("\nServer is listening");
 print("Waiting.......")
while True:
 conn, add = s.accept();
```

```
 with conn:
 print("Got connection from ",add);


 conn.send(b"You are Connected Please Select Your Choice:--\n1--
>Create client's


Account \n2-->Record Sales\n3-->Record Payments \n4-->Account
Overview\n");


 #1
data = conn.recv(1024); #2
choice=int(data.decode())
 data_sent="Please Enter Account No ..."#3
 conn.sendall(data_sent.encode());
 data = conn.recv(1024); #4
 acc_no=data.decode();
 if choice==1:
data_sent="Please Enter Name..."#1.1
conn.sendall(data_sent.encode());
 data = conn.recv(1024); #1.2
 Name=data.decode()
data_sent="Please Enter Balance..."#1.3
 conn.sendall(data_sent.encode());
 data = conn.recv(1024); #1.4
Amt=int(data.decode())
 names.append(Name)
account_no.append(acc_no)
 balance.append(Amt)


 flags.append(0)
 data_sent="Account Successfully Created"#1.5
 print("Account With "+acc_no+" Successfully Created")
conn.sendall(data_sent.encode());
elif choice==4:
 data_sent=str(balance[account_no.index(acc_no)])
conn.sendall(data_sent.encode());
print("Enquiry For "+acc_no+" Successfully Replied")
 elif choice==2 or choice==3 or choice==5:
time_stamp=local_time[0];
 print(local_time[0])
 local_time[0]+=1
print(local_time[0])
```

```
time_stamp=(str(time_stamp)).encode()
 conn.sendall(time_stamp);
 data_sent="Please Wait ..."

 conn.sendall(data_sent.encode());
req_timestamp.append(time_stamp)
print(req_timestamp)
action.append(choice)
 choices.append(choice)
ip_addresses.append(add[0])
print(str(ip_addresses))


 print("Transaction Request From "+acc_no+" Successfully Registered")
 s.close()
def reply_client(ip_address,choice):
 try:
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM);
except socket.error as err:
print("Could not setup socket");
 port = 12000;
 ip = ip_address
print("You "+ ip+" are selected")
with s:
 s.connect((ip,port));
rec = s.recv(1024);
 print(rec.decode());
 data_sent = str(choice)
 s.sendall(data_sent.encode())
choice=int(choice)
if choice==3:
 data_sent = "Please Enter Amount of Payment >>"
 s.sendall(data_sent.encode())
rec = s.recv(1024);
amt=int(rec.decode())
if balance[0]>=amt:


 balance[0]=balance[0]-amt
data_sent = "Transaction Successful Total Balance in
Account="+str(balance[0])
 print(balance[0])
 s.sendall(data_sent.encode())
else:
 data_sent="Insufficient Balance"
 s.sendall(data_sent.encode())
```

```
elif choice==2:
 data_sent = "Please Enter Amount of Sales >>"

s.sendall(data_sent.encode())
rec = s.recv(1024);
amt=int(rec.decode())
 balance[0]=balance[0]+amt
 data_sent = "Transaction Successful Total Amount="+str(balance[0])
s.sendall(data_sent.encode())
elif choice==5:
 data_sent = "Please Enter Account Number Of Receiver >>"
s.sendall(data_sent.encode())
 rec = s.recv(1024);
recv_acc_no=int(rec.decode())
 data_sent = "Please Enter Amount To Send >>"
s.sendall(data_sent.encode())
 rec = s.recv(1024);


 amt=int(rec.decode())
print (account_no)
index=account_no.index(str(recv_acc_no))
 if is_free(recv_acc_no)==False:
 data_sent = "Receiver Currently Unavailable"
s.sendall(data_sent.encode())
 return
else:
balance[index]=balance[index]+amt
 balance[0]=balance[0]-amt
 data_sent = "Transaction Successful Total Amount In Your
Account="+str(balance[0])
s.sendall(data_sent.encode())
 s.close()
curr_index=ip_addresses.index(ip)
 del ip_addresses[curr_index]
 del action[curr_index]
del choices[curr_index]
 del req_timestamp[curr_index]
if _name_ == '_main_':
 local_time.append(0)
 t1 = threading.Thread(target=serving)
 t2 = threading.Thread(target=richarts_algo)

t1.start()
```

```
 t2.start()
```

Client Side:

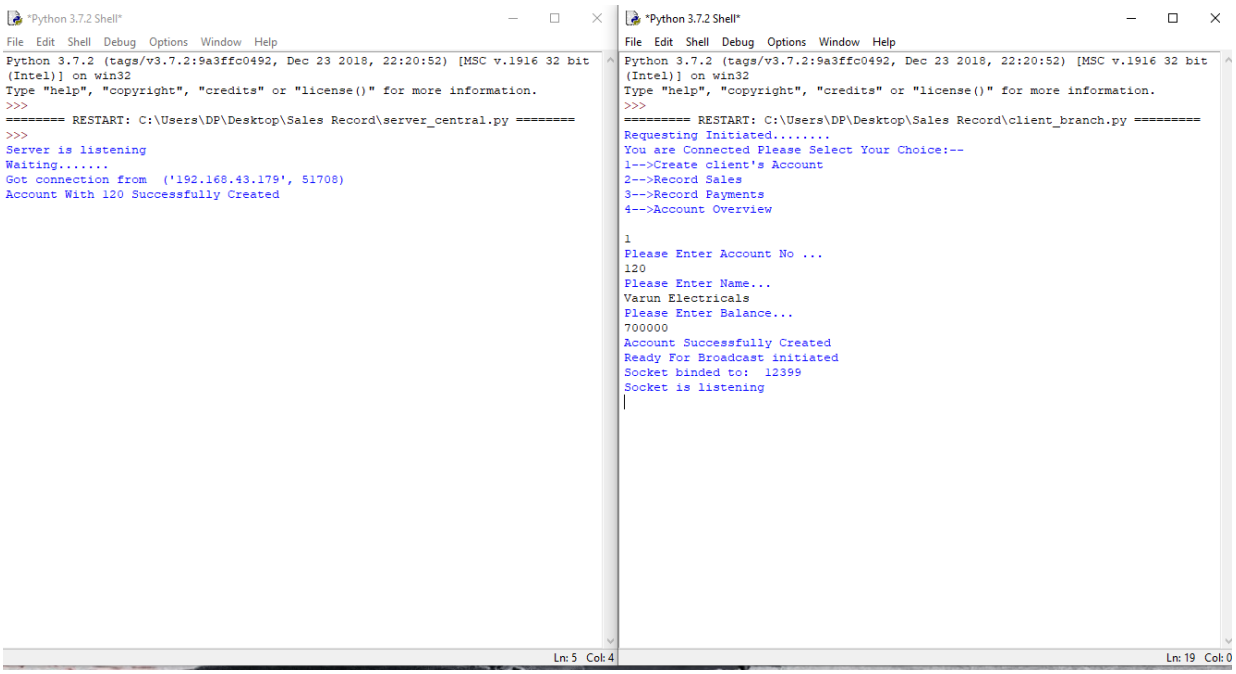
```
import socket;
import threading;
import time
acc_no=[] #integer
time_stamp=[] #integer
def request():
 try:
 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM);
 print("Requesting Initiated........");
except socket.error as err:
 print("Could not setup socket");
 port = 12365;
 ip ='192.168.43.179'
with s:
 s.connect((ip,port));
rec = s.recv(1024);#1
print(rec.decode());
 choice = input()
 data_sent = choice.encode();#2ENTER CHOICE
 s.sendall(data_sent);
choice=int(choice)
 rec = s.recv(1024);#3
 print(rec.decode());
 req_account = (input()).encode();#4 ENTER ACCOUNT NO
 s.sendall(req_account);
 acc_no.append(int(req_account))

if choice==1:
rec = s.recv(1024);#1.1
print(rec.decode());
Name=input()
data_sent = Name.encode();#1.2
 s.sendall(data_sent);
 rec = s.recv(1024);#1.3
 print(rec.decode());
 balance=input()
 data_sent = balance.encode();#1.4
 s.sendall(data_sent);
rec = s.recv(1024);#1.5
 print(rec.decode())
 elif choice==4:
rec = s.recv(1024);#1.3
```

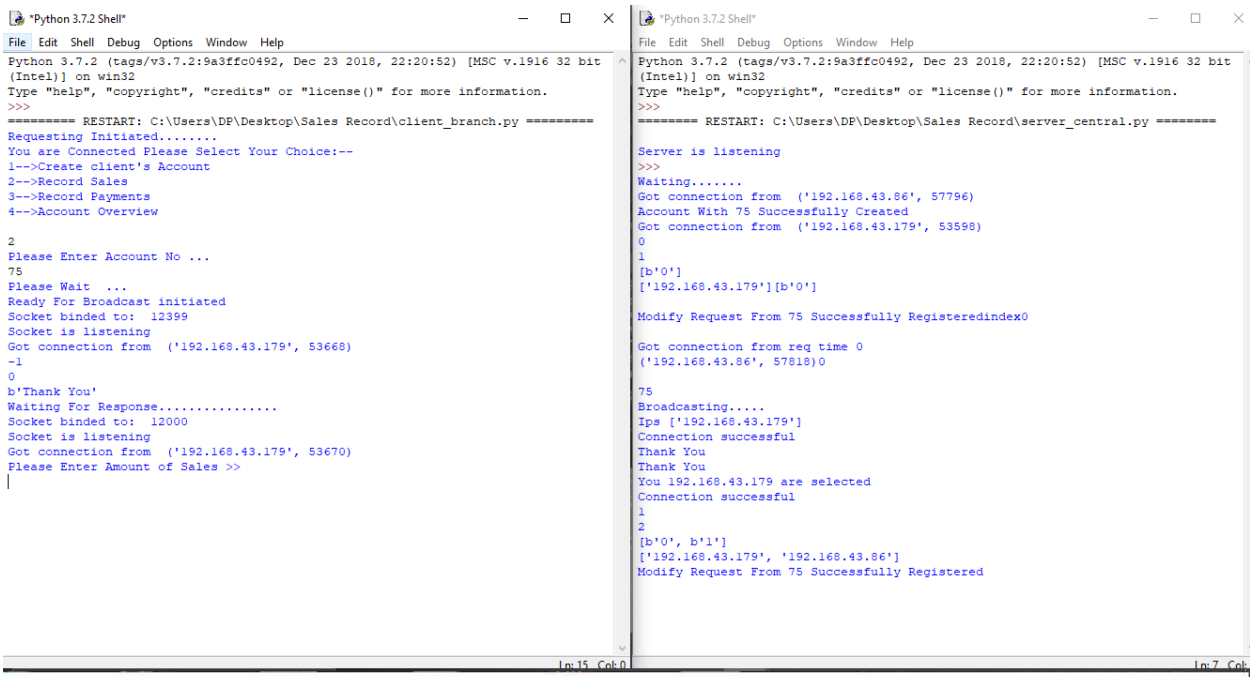
```
print(rec.decode())
elif choice==2 or choice==3 or choice==5:
 allocated_time_stamp=s.recv(1024) #storing allocated timestamp
 allocated_time_stamp=float(allocated_time_stamp.decode())
time_stamp.append(allocated_time_stamp)
 rec=s.recv(1024) ##printing wait message
print(rec.decode())
 acc_no.append(req_account) #
#print("Connection closed\n\n");
 s.close();
 #serving(int(choice))
def broad_cast_sender_receiver():
 try:
 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM);
print("Ready For Broadcast initiated");
 except socket.error as err:
 print("Socket creation failed");
port = 12399;
 s.bind(('', port));
 print("Socket binded to: ", port);
 s.listen(5);
 print("Socket is listening");

conn, add = s.accept();
 with conn:

print("Got connection from ",add);
conn.send(b"Connection successful");

 acc = conn.recv(1024);

 acc=acc.decode()
acc=int(acc)
tim=conn.recv(1024)
tim=tim.decode()
 tim=int(tim)
 print(str(acc))
 print(str(tim))
 if acc==-1:
 data_sent = ("Thank You").encode()
print(data_sent)
 conn.sendall(data_sent);
 execute()
 s.close();
print("Permission Granted")
 return;
```

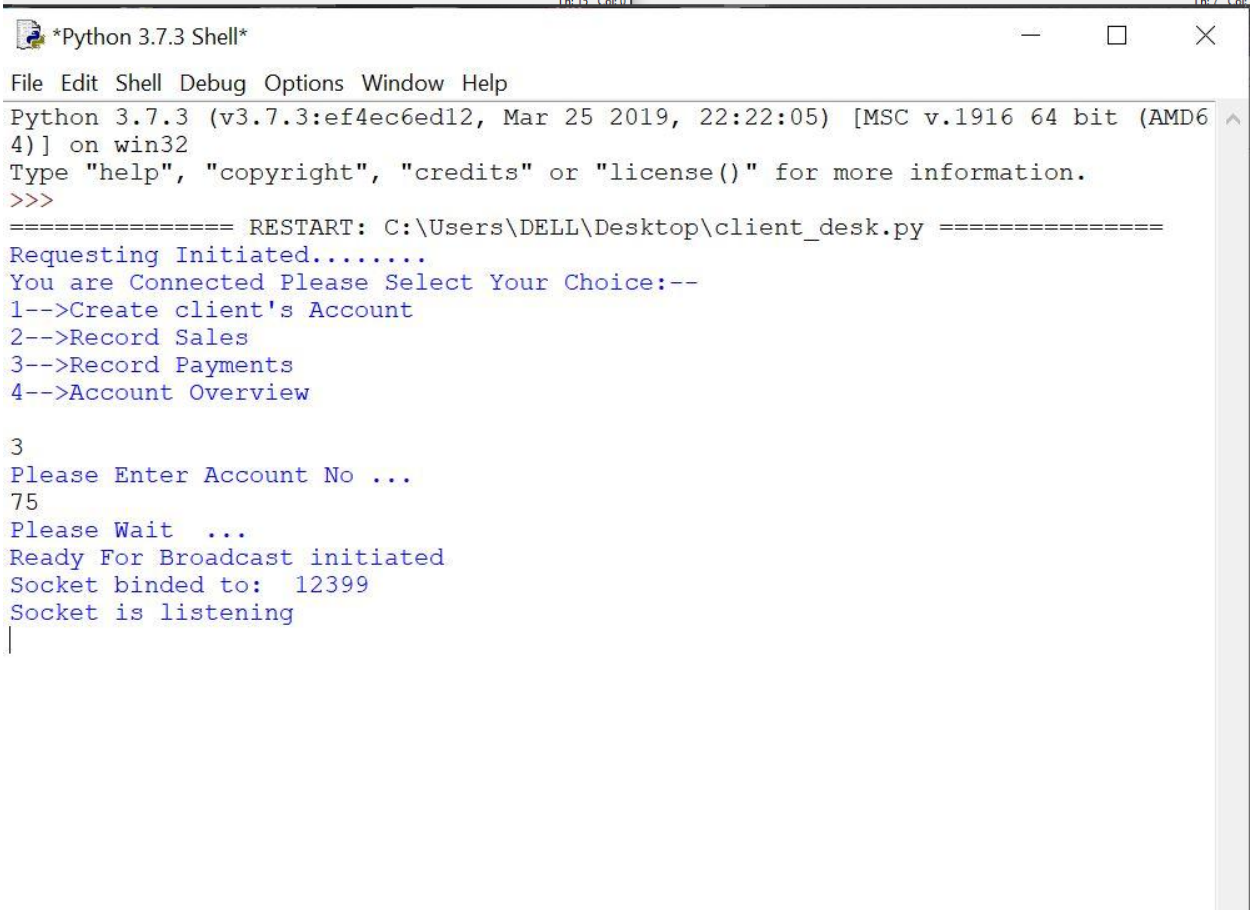
```
 elif acc_no[0]==acc and time_stamp[0]<=tim:
data_sent = (str(0)).encode()
print("cond 2")
 conn.sendall(data_sent);
 else:
 data_sent = (str(1)).encode()

 conn.sendall(data_sent);
 s.close();
def execute():
 try:
 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM);
print("Waiting For Response.............");
 except socket.error as err:
print("Socket creation failed");
 port = 12000;
 s.bind(('', port));
 print("Socket binded to: ", port);
 s.listen(5);
 print("Socket is listening");
while True:
 conn, add = s.accept();

with conn:
 print("Got connection from ",add);
 conn.send(b"Connection successful");
 data = conn.recv(1024);
 t=data.decode()
 #choice=int(t)
 if t=="3" or t=="2":
 data = conn.recv(1024);

 print(data.decode())
amount=input()
 data_sent=amount.encode()
 conn.sendall(data_sent)
 data = conn.recv(1024);
 print(data.decode())
 else:
 data = conn.recv(1024);
 print(data.decode())
 acc_no=input()
 data_sent=acc_no.encode()
conn.sendall(data_sent)
 data = conn.recv(1024);
 print(data.decode())
```

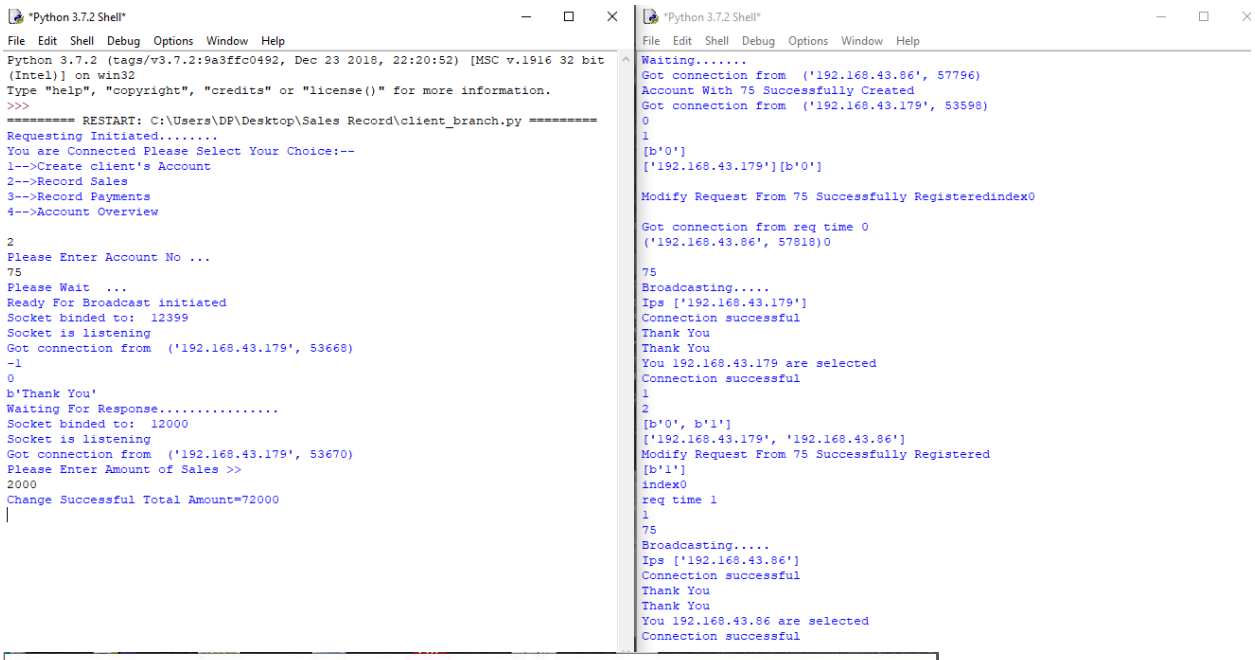
```
 amount=input()
 data_sent=amount.encode()
 conn.sendall(data_sent)
data = conn.recv(1024);
 print(data.decode())
 s.close();
def requesting():
 request()
def executing():
 execute()

def broadcasting():
 broad_cast_sender_receiver()
if _name_ == '_main_':

requesting()

broadcasting()
```

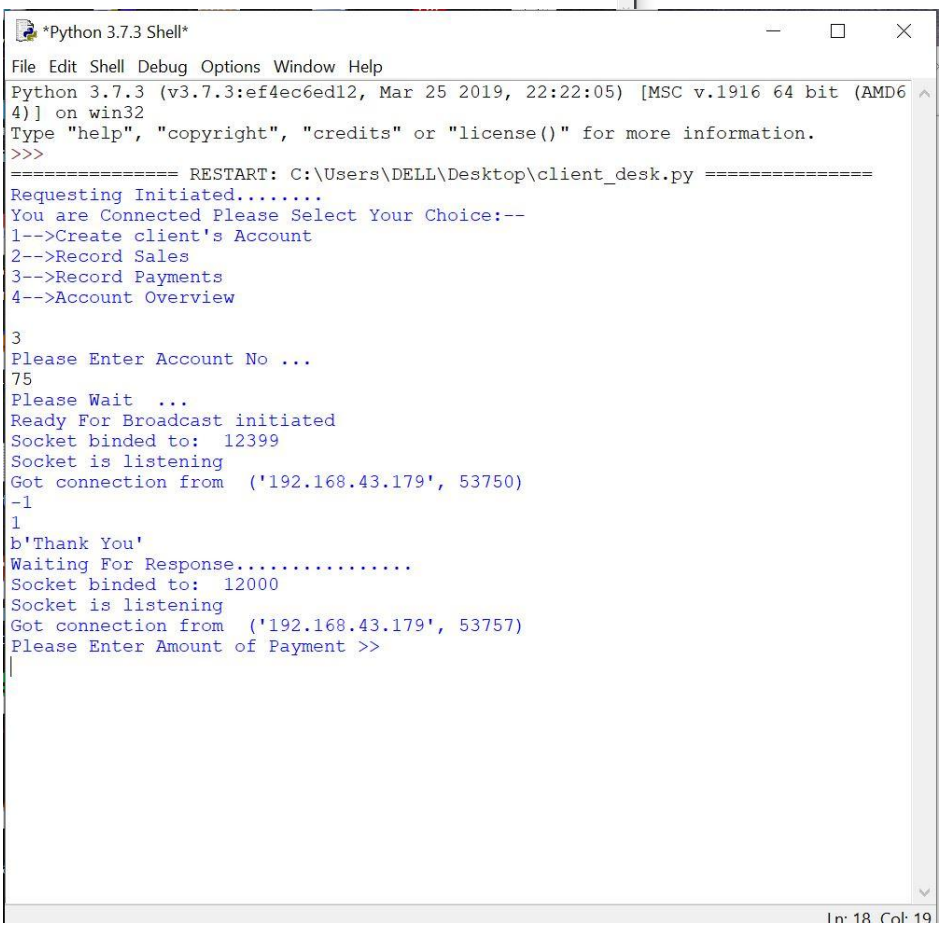
IV. RESULTS / OUTPUT:

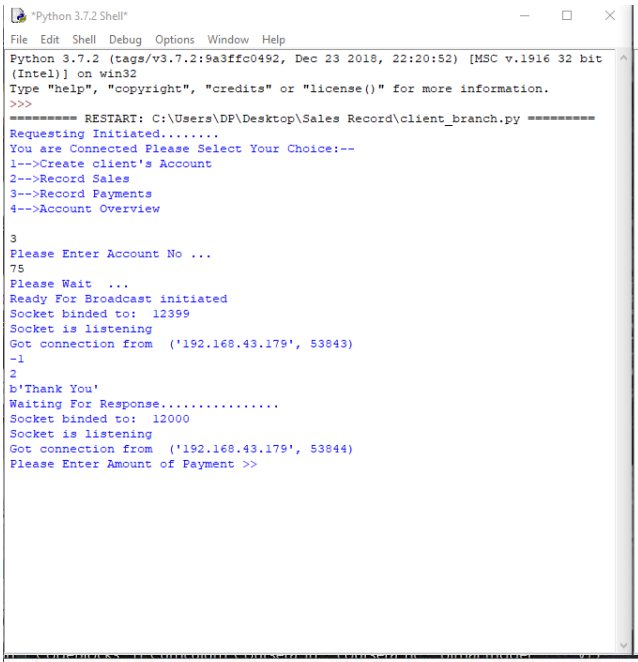



Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======== RESTART: C:\Users\DP\Desktop\Sales Record\client_branch.py ========
Requesting Initiated.....
You are Connected Please Select Your Choice:--
1-->Create client's Account
2-->Record Sales
3-->Record Payments
4-->Account Overview
2
Please Enter Account No ...
75
Please Wait ...
Ready For Broadcast initiated
Socket binded to: 12399
Socket is listening
Got connection from ('192.168.43.179', 53668)
-1
0
b'Thank You'
Waiting For Response.....
Socket binded to: 12000
Socket is listening
Got connection from ('192.168.43.179', 53670)
Please Enter Amount of Sales >>
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======== RESTART: C:\Users\DP\Desktop\Sales Record\server_central.py ========
Server is listening
>>>
Waiting.....
Got connection from ('192.168.43.86', 57796)
Account With 75 Successfully Created
Got connection from ('192.168.43.179', 53598)
0
1
[b'0']
['192.168.43.179'][b'0']
Modify Request From 75 Successfully Registeredindex0
Got connection from req time 0
('192.168.43.86', 57818)0
75
Broadcasting.....
Ips ['192.168.43.179']
Connection successful
Thank You
Thank You
You 192.168.43.179 are selected
Connection successful
1
2
[b'0', b'1']
['192.168.43.179', '192.168.43.86']
Modify Request From 75 Successfully Registered

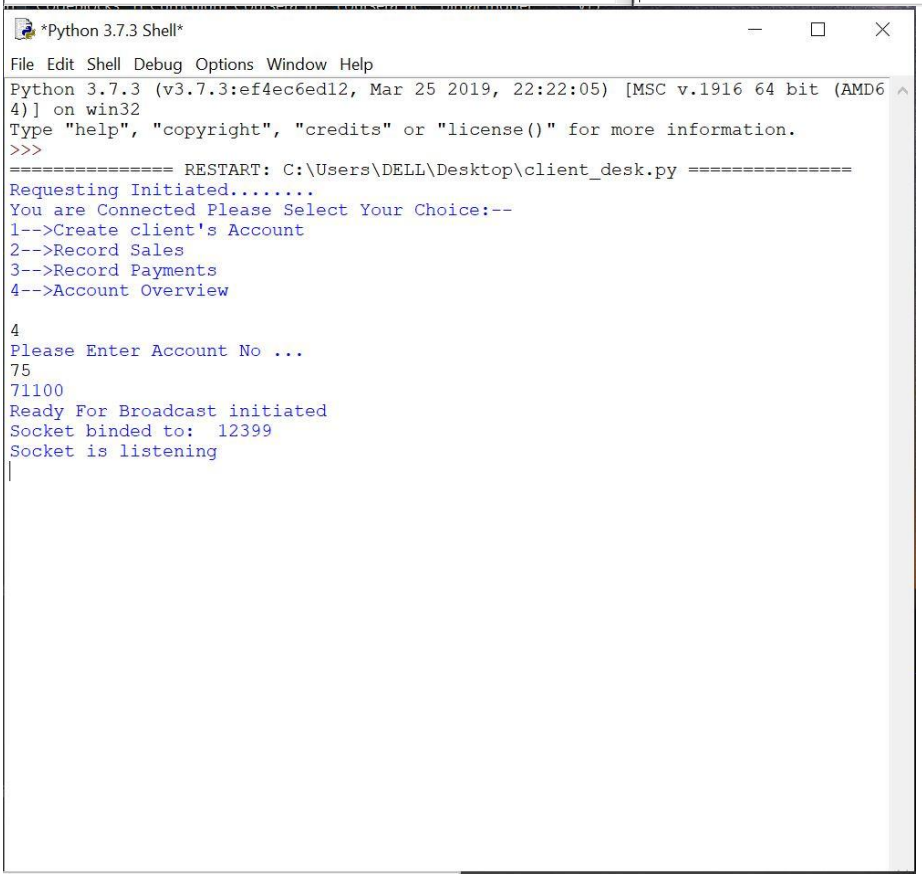

Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Users\DELL\Desktop\client_desk.py ==========
Requesting Initiated.....
You are Connected Please Select Your Choice:--
1-->Create client's Account
2-->Record Sales
3-->Record Payments
4-->Account Overview
3
Please Enter Account No ...
75
Please Wait ...
Ready For Broadcast initiated
Socket binded to: 12399
Socket is listening


Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========= RESTART: C:\Users\DP\Desktop\Sales Record\client_branch.py =========
Requesting Initiated........
You are Connected Please Select Your Choice:--
1-->Create client's Account
2-->Record Sales
3-->Record Payments
4-->Account Overview
2
Please Enter Account No ...
75
Please Wait ...
Ready For Broadcast initiated
Socket binded to: 12399
Socket is listening
Got connection from ('192.168.43.179', 53668)
-1
0
b'Thank You'
Waiting For Response.....
Socket binded to: 12000
Socket is listening
Got connection from ('192.168.43.179', 53670)
Please Enter Amount of Sales >>
2000
Change Successful Total Amount=72000
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Waiting.....
Got connection from ('192.168.43.86', 57796)
Account With 75 Successfully Created
Got connection from ('192.168.43.179', 53598)
0
1
[b'0']
['192.168.43.179'][b'0']
Modify Request From 75 Successfully Registeredindex0
Got connection from req time 0
('192.168.43.86', 57818)0
75
Broadcasting.....
Ips ['192.168.43.179']
Connection successful
Thank You
Thank You
You 192.168.43.179 are selected
Connection successful
1
2
[b'0', b'1']
['192.168.43.179', '192.168.43.86']
Modify Request From 75 Successfully Registered
[b'1']
index0
req time 1
1
75
Broadcasting.....
Ips ['192.168.43.86']
Connection successful
Thank You
Thank You
You 192.168.43.86 are selected
Connection successful


Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Users\DELL\Desktop\client_desk.py ==========
Requesting Initiated........
You are Connected Please Select Your Choice:--
1-->Create client's Account
2-->Record Sales
3-->Record Payments
4-->Account Overview
3
Please Enter Account No ...
75
Please Wait ...
Ready For Broadcast initiated
Socket binded to: 12399
Socket is listening
Got connection from ('192.168.43.179', 53750)
-1
1
b'Thank You'
Waiting For Response.....
Socket binded to: 12000
Socket is listening
Got connection from ('192.168.43.179', 53757)
Please Enter Amount of Payment >>
Ln: 18 Col: 19


Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========= RESTART: C:\Users\DP\Desktop\Sales Record\client_branch.py =========
Requesting Initiated.....
You are Connected Please Select Your Choice:--
1-->Create client's Account
2-->Record Sales
3-->Record Payments
4-->Account Overview
3
Please Enter Account No ...
75
Please Wait ...
Ready For Broadcast initiated
Socket binded to: 12399
Socket is listening
Got connection from ('192.168.43.179', 53843)
-1
2
b'Thank You'
Waiting For Response.....
Socket binded to: 12000
Socket is listening
Got connection from ('192.168.43.179', 53844)
Please Enter Amount of Payment >>


Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
You 192.168.43.179 are selected
Connection successful
1
2
[b'0', b'1']
['192.168.43.179', '192.168.43.86']
Modify Request From 75 Successfully Registered
[b'1']
index0
req time 1
1
75
Broadcasting.....
Ips ['192.168.43.86']
Connection successful
Thank You
Thank You
You 192.168.43.86 are selected
Connection successful
71100
Got connection from ('192.168.43.179', 53823)
2
3
[b'2']
['192.168.43.179'][b'2']
Modify Request From 75 Successfully Registeredindex0
Got connection from req time 2
('192.168.43.86', 57841)2
75
Broadcasting.....
Ips ['192.168.43.179']
Connection successful
Thank You
Thank You
You 192.168.43.179 are selected
Connection successful


Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Users\DELL\Desktop\client_desk.py ==========
Requesting Initiated.....
You are Connected Please Select Your Choice:--
1-->Create client's Account
2-->Record Sales
3-->Record Payments
4-->Account Overview
4
Please Enter Account No ...
75
71100
Ready For Broadcast initiated
Socket binded to: 12399
Socket is listening

V. CONCLUSION AND FUTURE WORK:

Ricart Agrawala can be extended to work on practical network applications. The correctness of the Ricart Agrawala Algorithm should not be affected when inserting new nodes into the network. There can be a practical network insertion of new nodes. Whenever new nodes are added it should be able to update the sequence number, request received and the requests it is going to reply back or acknowledge. Ricart Agrawala can be extended to solve Dining Philosopher's Problem where there are several sites and several numbers of processes working in each site. The demonstration that we have presented here is one practical application of the algorithm. The algorithm if fixed, and deployed correctly has many applications in the industry such as banking sector, reservation systems, etc.

VI. REFERENCES:

1) https://www.geeksforgeeks.org/ricart-agrawala-algorithm-in-mutual-exclusion-in-distributed-system/
2) https://www.geeksforgeeks.org/lamports-algorithm-for-mutual-exclusion-in-distributed-system/
3) https://www.cs.uic.edu/~ajayk/Chapter9.pdf
4) https://www.geeksforgeeks.org/ricart-agrawala-algorithm-in-mutual-exclusion-in-distributed-system/
5) https://link.springer.com/chapter/10.1007/3-540-44450-5_26
6) https://ieeexplore.ieee.org/abstract/document/508024/
7) https://ieeexplore.ieee.org/abstract/document/1183620/
8) https://www.geeksforgeeks.org/mutual-exclusion-in-distributed-system/
9) https://en.wikipedia.org/wiki/Ricart%E2%80%93Agrawala_algorithm
10) https://en.wikipedia.org/wiki/Lamport%27s_distributed_mutual_exclusion_algorithm
11) https://ieeexplore.ieee.org/abstract/document/7030782/