Ayush Goyal
190905522 CSE D 62

Distributed Systems Week 7: Lab 7: Election Algorithms

Lab Exercises:

Q1) Simulate a scenario in distributed systems to implement the Bully Algorithm for choosing a coordinator node amongst the participative nodes of the system after the collapse of the existing coordinator node in the system.

Code:

"bully.py"

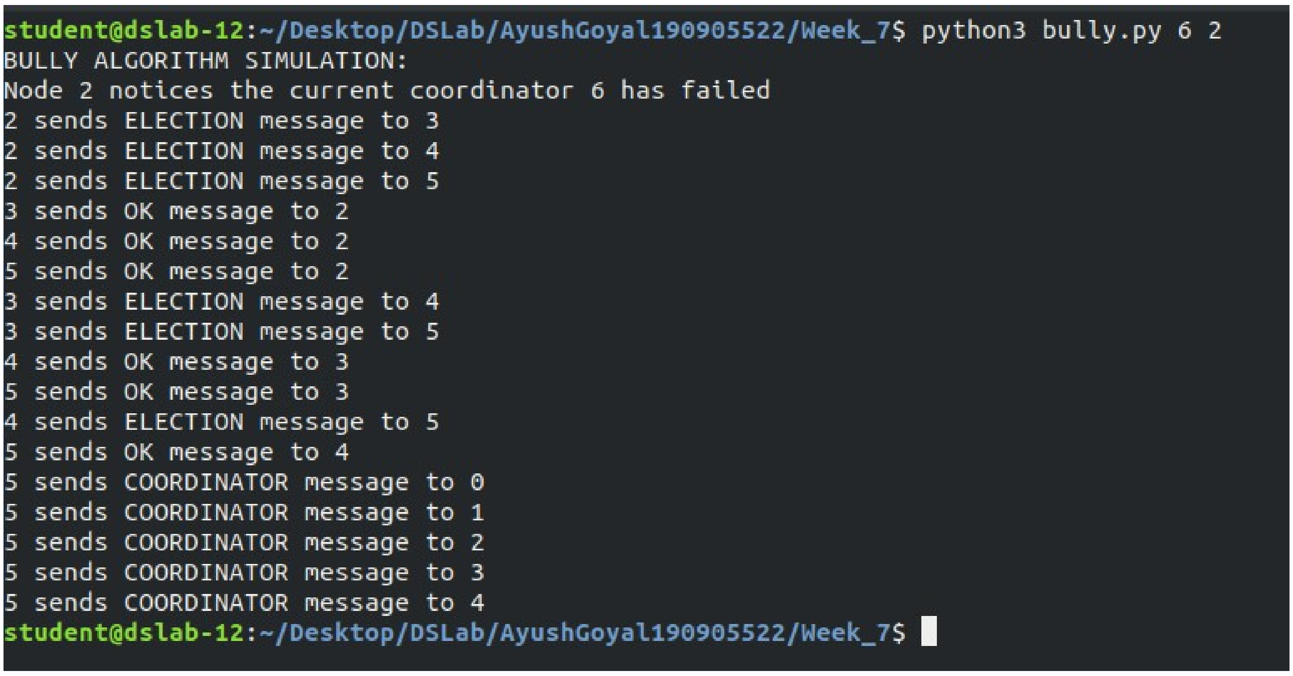
```
import sys

noOfNodes = int(sys.argv[1])
initiatorNode = int(sys.argv[2])

def bully_algorithm():
    print("BULLY ALGORITHM SIMULATION:")
    print('Node %s notices the current coordinator %s has failed' % (initiatorNode, noOfNodes))
    biggerNodes = []
    for i in range(initiatorNode+1, noOfNodes):
        print("%s sends ELECTION message to %s" % (initiatorNode,i))
        biggerNodes.append(i)
    for i in biggerNodes:
        print("%s sends OK message to %s" % (i, initiatorNode))
    while len(biggerNodes) != 1:
        i = biggerNodes[0]
        for j in range(i+1, noOfNodes):
            print("%s sends ELECTION message to %s" % (i, j))
        for k in range(i+1, noOfNodes):
            print("%s sends OK message to %s" % (k, i))
        biggerNodes.remove(i)
    newCoordinatorNode = biggerNodes[0]
    for i in range(0, newCoordinatorNode):
        print("%s sends COORDINATOR message to %s" %(newCoordinatorNode, i))

if __name__ == '__main__':
    bully_algorithm()
```

Output:



```
student@dslab-12:~/Desktop/DSLab/AyushGoyal190905522/Week_7$ python3 bully.py 6 2
BULLY ALGORITHM SIMULATION:
Node 2 notices the current coordinator 6 has failed
2 sends ELECTION message to 3
2 sends ELECTION message to 4
2 sends ELECTION message to 5
3 sends OK message to 2
4 sends OK message to 2
5 sends OK message to 2
3 sends ELECTION message to 4
3 sends ELECTION message to 5
4 sends OK message to 3
5 sends OK message to 3
4 sends ELECTION message to 5
5 sends OK message to 4
5 sends COORDINATOR message to 0
5 sends COORDINATOR message to 1
5 sends COORDINATOR message to 2
5 sends COORDINATOR message to 3
5 sends COORDINATOR message to 4
student@dslab-12:~/Desktop/DSLab/AyushGoyal190905522/Week_7$
```

Q2) Simulate a scenario in distributed systems to implement the Ring Algorithm for choosing a coordinator node amongst the participative nodes of the system after the collapse of the existing coordinator node in the system.

Code:

“ringserver.py”

```
import sys
import threading
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = socket.gethostname()
port = 7777
try:
   s.bind((host, port))
except socket.error as msg:
   print("bind failed" + str(msg))
   sys.exit()
s.listen(10)
process_sockets_list = []
process_list = []
neighbor_list = []
msg_token = ""
def recv_message(conn):
    while True:
        try:
            received = conn.recv(1024)
            msg_token = received.decode('utf-8')
            print("received token: " + msg_token)
        except:
            continue
        if "Coordinator: " in msg_token :
            le=msg_token.split()
            leader=le[1]
        process_index = process_sockets_list.index(conn)
        if len(process_sockets_list)==process_index+1 :
            to_process=0
        else :
            to_process=process_index+1
        try :
            process_sockets_list[to_process].send(received)
            print("sending :" + received.decode('utf-8'))
        except :
            if process_list[to_process]!=leader :
                process_sockets_list[to_process+1].send(received)
                print("sending :" + received.decode('utf-8'))
            process_sockets_list[to_process].close()
            process_sockets_list.remove(process_sockets_list[to_process])
            process_list.remove(process_list[to_process])
            continue

while True:
    try:
        connection, addr = s.accept()
        process_sockets_list.append(connection)
        recv_process_id = connection.recv(1024)
        from_to_process = recv_process_id.decode('utf-8')
        process_list.append(from_to_process)
        print("Process: " + from_to_process)
        start_thread = threading.Thread(target=recv_message, args=(connection,))
        start_thread.start()
    except socket.error as msg:
        print("thread failed"+msg)

connection.close()
s.close()
```

“ringclient.py”

```
import socket
import threading
import time
import sys
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

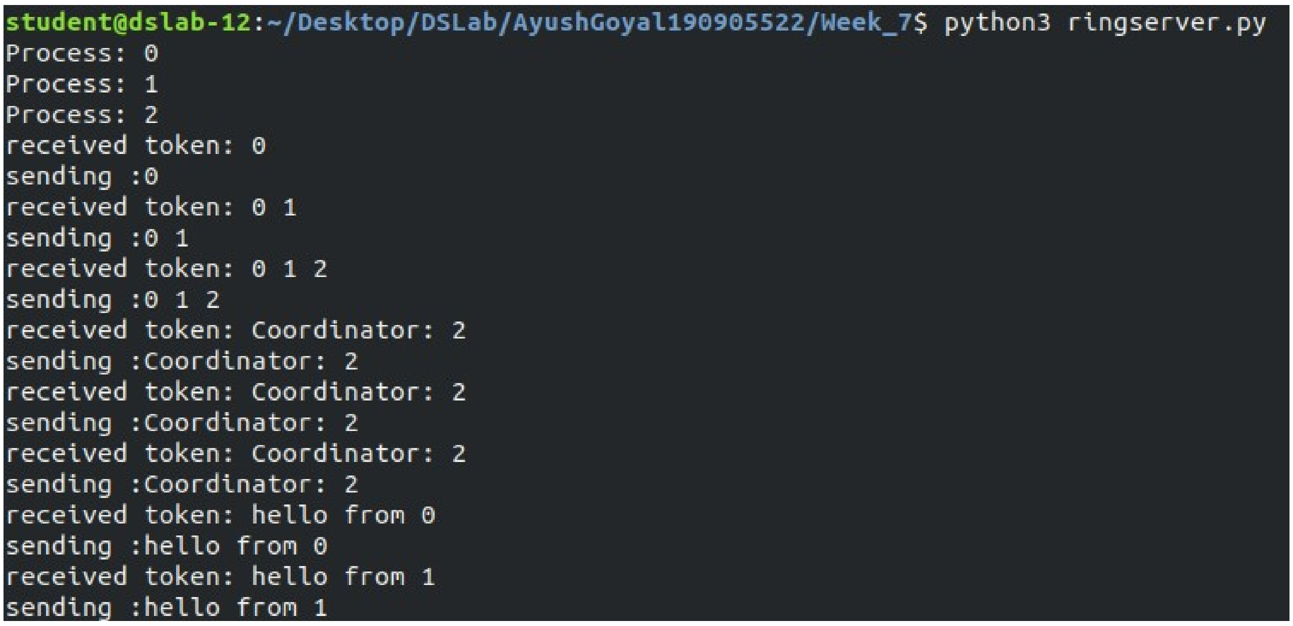
```
host = socket.gethostname()
to_port = 7777
s.connect((host, to_port))
my_id = str(sys.argv[1])
s.send(my_id.encode('utf-8'))
leader="-1"
def initiate_election(s):
    time.sleep(1)
    s.send(my_id.encode('utf-8'))
    print("token sent: " + my_id)
    print("Election initiated")


def Ring_Election_Algorithm(s):
    while True:
        global leader
        try:
            s.settimeout(15)
            received = s.recv(1024)
            s.settimeout(None)
            received_token_list = received.decode('utf-8')
        except socket.timeout:
            leader = "0"
            initiate_election(s)
            continue
        if my_id in received_token_list and "Coordinator: " not in received_token_list and "hello"
not in received_token_list:
            leader = max(received_token_list)
            forwarding_leader = "Coordinator: " + leader
            time.sleep(1)
            s.send(forwarding_leader.encode('utf-8'))
        elif my_id not in received_token_list and "Coordinator: " not in received_token_list and
"hello" not in received_token_list :
            print("rec tok: " + received_token_list)
            leader = "0"
            received_token_list = received_token_list + " " + my_id
            time.sleep(1)
            s.send(received_token_list.encode('utf-8'))
            print("adding token: " + received_token_list)
        elif ("hello" in received_token_list or "Coordinator: " in received_token_list )and leader=="-
1":
            leader="0"
            initiate_election(s)
        elif "Coordinator: " in received_token_list and leader not in received_token_list :
            print(received_token_list)
            le=received_token_list.split()
            leader=le[1]
            time.sleep(1)
            s.send(received_token_list.encode('utf-8'))
        else :
            if leader=="-1" or leader=="0":
                continue
            else :
                print(received_token_list)
                communicate = "hello" + " from " + my_id
                time.sleep(1)
                s.send(communicate.encode('utf-8'))
                continue


recv_thread = threading.Thread(target=Ring_Election_Algorithm, args=(s,))
recv_thread.start()
recv_thread.join()
s.close()
```

<u>Output:</u>

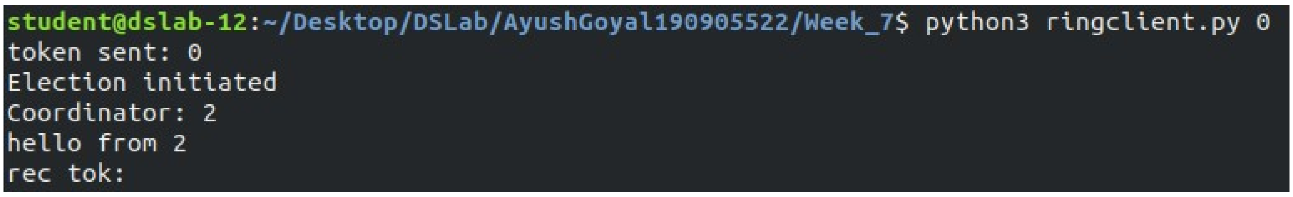
Server Ouput:



```
student@dslab-12:~/Desktop/DSLab/AyushGoyal190905522/Week_7$ python3 ringserver.py
Process: 0
Process: 1
Process: 2
received token: 0
sending :0
received token: 0 1
sending :0 1
received token: 0 1 2
sending :0 1 2
received token: Coordinator: 2
sending :Coordinator: 2
received token: Coordinator: 2
sending :Coordinator: 2
received token: Coordinator: 2
sending :Coordinator: 2
received token: hello from 0
sending :hello from 0
received token: hello from 1
sending :hello from 1
```

Client 1 Output:

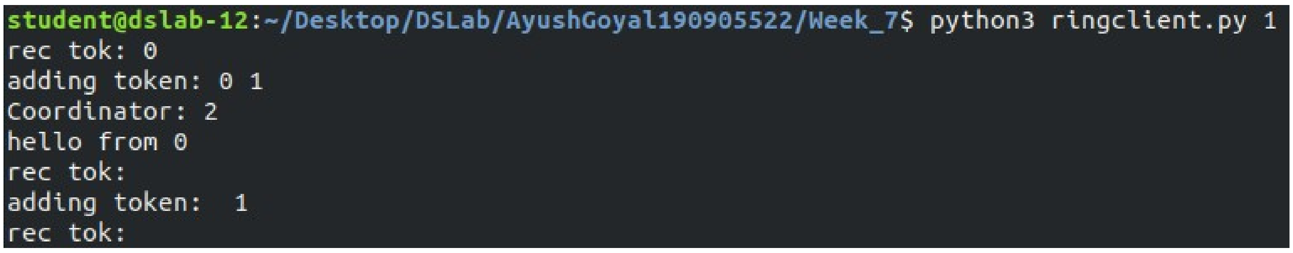
python3 ringclient.py 0



```
student@dslab-12:~/Desktop/DSLab/AyushGoyal190905522/Week_7$ python3 ringclient.py 0
token sent: 0
Election initiated
Coordinator: 2
hello from 2
rec tok:
```

Client 2 Output:

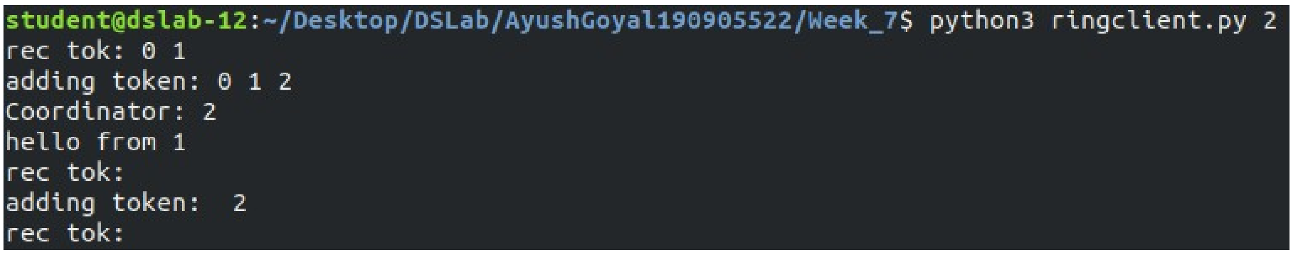
python3 ringclient.py 1



```
student@dslab-12:~/Desktop/DSLab/AyushGoyal190905522/Week_7$ python3 ringclient.py 1
rec tok: 0
adding token: 0 1
Coordinator: 2
hello from 0
rec tok:
adding token:  1
rec tok:
```

Client 3 Output:

python3 ringclient.py 2



```
student@dslab-12:~/Desktop/DSLab/AyushGoyal190905522/Week_7$ python3 ringclient.py 2
rec tok: 0 1
adding token: 0 1 2
Coordinator: 2
hello from 1
rec tok:
adding token:  2
rec tok:
```

THE END