CESC 327
Assignment 2: Distributed Chat

Professor Oscar Morales

Due date March 7

1 Objectives and Project Description

The objectives of this project are:
1) Understand the use of client and server sockets in Java
2) Identify the advantages of overlay networks
3) Differentiate the role of server and client
4) identify the advantages of JSON
5) Integrate threads in network programming

The task is to develop a distributed chat without a central coordinators. The processors will form an overlay topology known as a ring. In other words, every processor i will keep only the successor $succ(i)$ and $pred(i)$ as shown in Figure 3.

When a user i wants to talk with a friend j it sends a PUT message to the successor. PUT consists of the source alias, the destination alias as well as the message. We assume that each user knows the alias of the friends.

A user that wants to join the system sends a $JOIN$ message to a current participant, say i (if there is any participant, it will act as the only participant). Let j be the process that send the $JOIN$ to i and let $i+1$ be the successor of i. When i receives the request, it updates its routing table by replacing its predecessor with j and response with an $ACCEPT$ message that contains the ip and port of the previous predecessor node in the ring.

When a user wants to leave the room, it sends a $LEAVE$ message to the successor with the ip and port of the predecessor so that the nodes can reconstruct the ring.

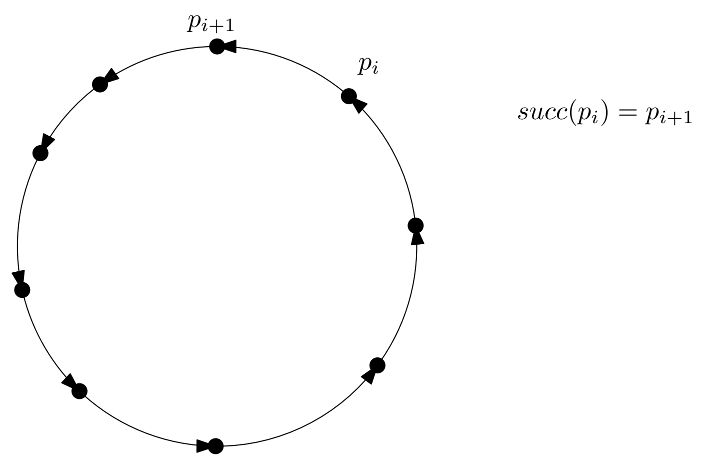

Figure 1: Ring

2 Messages

- *PUT*: $\langle alias_{sender}, alias_{receiver}, text\rangle$: Flood the message along the ring.
  $alias_{sender}$: initiation alias
  $id_{receiver}$: destination alias
  text: Message to be sent.

  The message is flooded until it reaches either $alias_{receiver}$ or $alias_{sender}$. If the message arrives at $alias_{sender}$, then the user was not available.

- *JOIN*: $\langle port\rangle$: port where the sender is listening.
  port: port of the sender. The peer replies with *ACCEPT* to the new member with the values $ip_{pred}, port_{pred}$ of the previous node and then update ip_{pred} with the ip obtained from the socket connection and $port_{pred}$ with received *port*.

  The peer also replies with *ACCEPTED* to the previous peer with the new values in $ip_{pred}, port_{pred}$.

- *ACCEPT*: $\langle ip_{pred}, port_{pred}\rangle$.
  ip_{pred}: IP of the previous node in the rings.
  $port_{pred}$: port of the previous node in the rings.

  When the new member receives it, it updates its $ip_{pred}, port_{pred}$ with the received ones and accordingly sets $ip_{suc}, port_{suc}$.

Joining Protocol

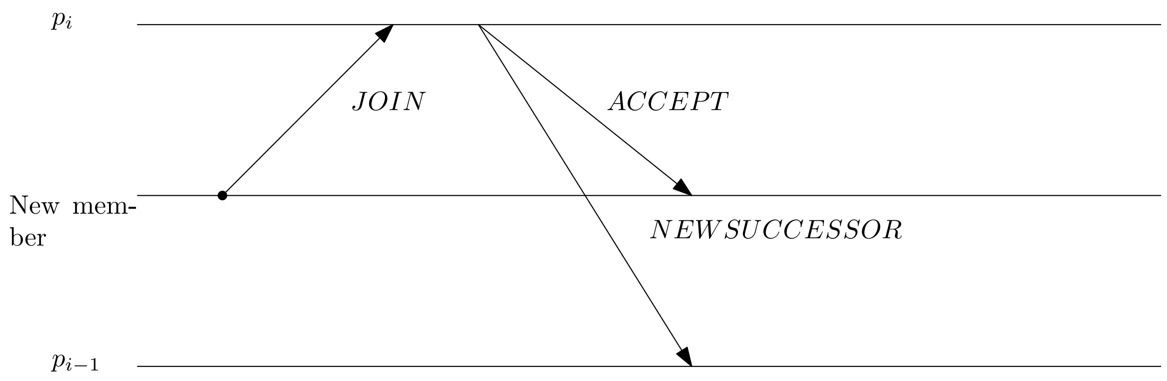

Figure 2: Ring

- $NEWSUCCESSOR$: $\langle ip_{succ}, port_{succ} \rangle$.
- $LEAVE$: $\langle ip_{pred}, port_{pred} \rangle$. It updates its routing table with the received information and sends $NEWSUCCESSOR$ to the predecessor.

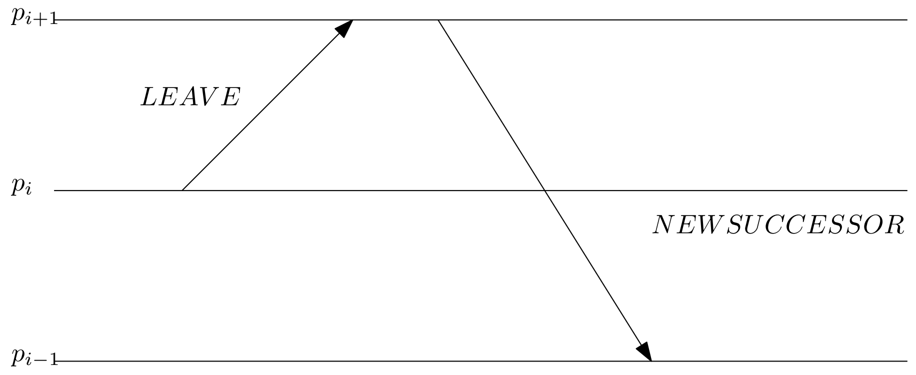

Figure 3: Ring

3 Flooding

The communication is based on a flooding protocol.

```
PROTOCOL Flooding.
   Status Values: S = {INITIATOR, IDLE, DONE};
   $S_{INIT} = {INITIATOR, IDLE};
   $S_{TERM} = {DONE}$
   Restrictions:Total Reliability,  Connectivity, and
                                Unique Initiator.

INITIATOR
   Spontaneously
   begin
       send(M) to N(x);
       become DONE;
   end
IDLE
   Receiving(I)
   begin
       Process(M);
       send(M) to N(x) - {sender};
       become DONE;
   end
```

4 Grading

Criteria	Weight
Documentation of your program	15%
Source code (good modularization, coding style, comments)	15%
Execution output	50%
Use of mutex	10%
Originality	10%

The documentation must be generated using Doxygen or Javadocs.

5 Deliverables

Compress in a zip file and upload to beachboard. The zipfile must contain to folders:
1.- Src: All the source code
2.- Docs: HTML with the documentation. It has to contain the definitions of the methods with a short description, parameters and output of the method.