DS Project - Report

Group members

IIB2019002 PRADHUMAN SINGH BAID
IIB2019005 SANDEEP KUMAR
IIB2019008 SHYAM TAYAL
IIB2019009 ABHIJEET SONKAR
IIB2019010 AVNEESH KUMAR

Introduction

In Distributed Systems, there are a lot of algorithms that are used in order to facilitate and optimize the computation. We have developed a chat application. And specifically used distributed System concepts in order to simulate two of the Distributed Computing control algorithms namely: The leader Election (Chang Roberts) Algorithm and Distributed Mutual Exclusion (Ring Based) Algorithm. The chat system has been designed in accordance with the assignment, The chat system allows each user to read a text file that includes the number of users in the system and also describes a distributed execution represented as a space-time diagram. The system has the provision to generate this text file randomly based on the number of users given as input.

For developing this system we have used python socket programming to enable peer-to-peer connection, and this application can be run using a command-line interface.

Algorithms

Leader Election (Chang and Roberts) Algorithm:

The algorithm assumes that each process in the network is assigned a Unique ID and processes are arranged in a unidirectional ring. Initially, each process is the initiator. All initiators are colored RED. Non-initiators are colored as BLACK.

Initiate:

```
Each initiator process i
      Send out token <i> to next process in ring
```

After sending token:

```
For each initiator process i:
      If token <j> is received and j < i
            Do nothing
      Else if token <j> is received and j > i
            Send token <j> to next process in ring
            Color process i as BLACK
      Else if token <j> is received and j==i
            L(i) = i
            You are leader.
            Send Leader Message to every other process.
```

```
For each Non-initiator process i:
      If token <j> is received
            Send token <j> to next process in ring
```

Assuming there are no failures this algorithm will finish. The algorithm works for any number of processes N, and does not require any process to know how many processes are in the ring.

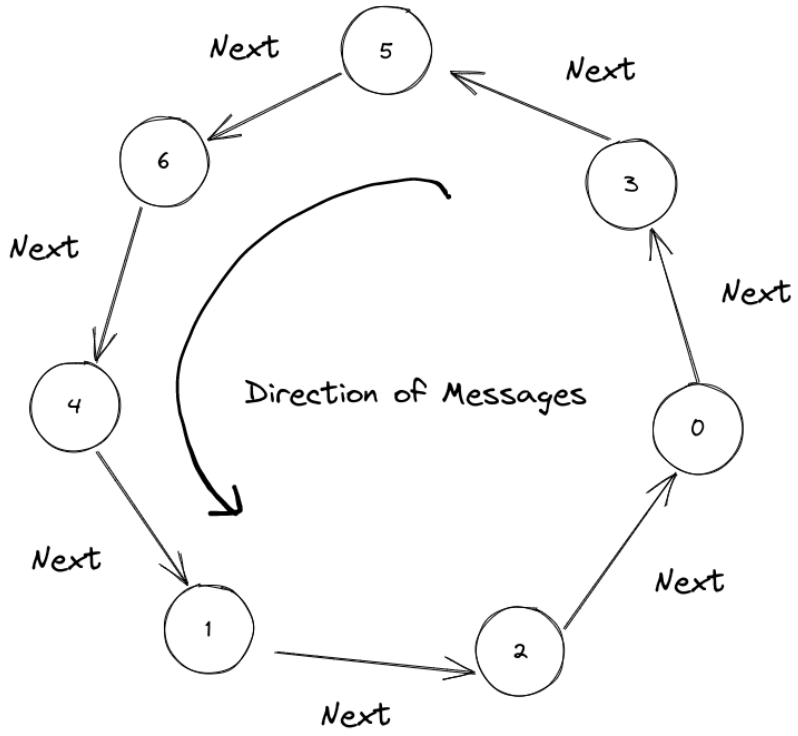


Fig. 1

Distributed Mutual Exclusion (Ring based) Algorithm:

This algorithm assumes that there are some processes that wish to execute the critical sections. We assume that processes are arranged in a ring form and there is only one token. Any process Pi holding token has control to execute critical section. The rest of the processes must wait in order to receive token and execute the critical section. At the very start, we give token to process with the smallest ID.

```
For every process
      If TOKEN received
            If want to execute CS
                  Execute CS
                  Send TOKEN to next Process in ring
            If DO NOT want to execute CS
                  Send TOKEN to next process in ring
      Else
            Wait for TOKEN
```

So, on receiving a token, if the process wishes to execute the critical section, can start executing CS. else it can pass the token to its successor in the ring.

Only one process has the token, and hence the lock on the resource, at a time. Therefore, mutual exclusion is guaranteed. Order is also well-defined, so starvation cannot occur. The biggest drawback of this algorithm is that if a token is lost, it will have to be generated. Determining that a token is lost can be difficult.

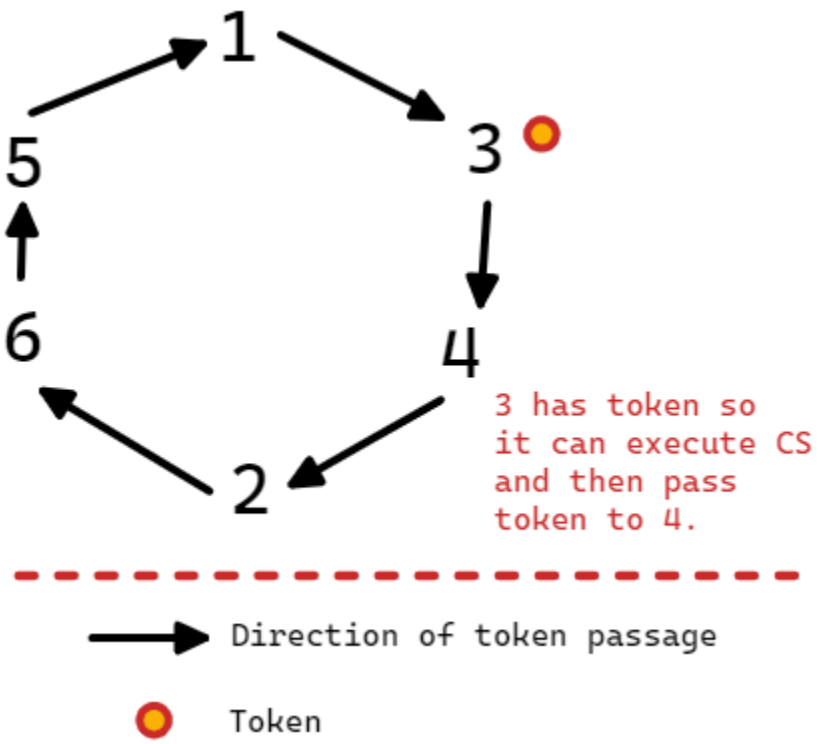


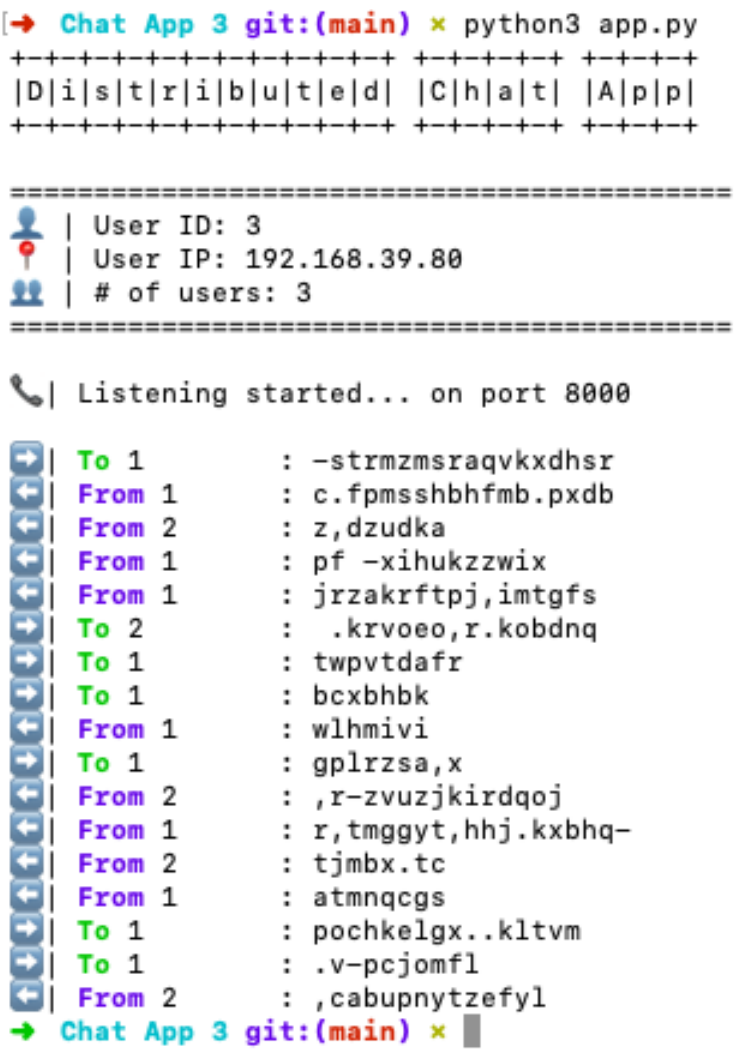
Fig. 2

Progress

Part 1:

- **Input File generator for Chat app:**
  - Python Script (takes command-line arguments: # of users, # of messages, respective IP addresses) for generating a random text file as an input for all the users.
  - Input File Description: This contains details including the number of users, their IP addresses, and ordered messages with their sending and receiving entities.

- **Chat App :**
  - Python script for enabling remote between different users on a network
  - Takes input text file as input and automates the message sending and receiving part.
  - Script description: Uses socket programming for establishing a connection, and sending and receiving messages. For this, we used the **socket** library.
  - Uses queue data structure for synchronizing transactions (sending/receiving messages).
  - The chat app simulates the space-time diagram represented by the input file.



Part 2 :

- **Input File generator for leader election algorithm :**
  - Similarly produces a text file containing information including the number of users, their respective IP addresses and a string of randomly assigned user IDs.
  - The file also contains a unidirectional ring of nodes in the system. This ring is needed as we implemented **Chang-Roberts Algorithm.**

- **Leader Election App :**
  - Python Script that implements Chang-Roberts Algorithm on a unidirectional ring.
  - All the specifications remain the same as the Chat App.
  - The script automatically runs and finds the leader among all the nodes in the system.
  - After the leader is found it broadcasts the message to all the nodes about its role as leader**.**

```
+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+-+ +-+-+ +-+-+-+-+-+-+-+-+-+-+-+-+-+-+ +-+-+-+-+
|L|e|a|d|e|r| |E|l|e|c|t|i|o|n| |i|n| |U|n|i|d|i|r|e|c|t|i|o|n|a|l| |R|i|n|g|
+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+-+ +-+-+ +-+-+-+-+-+-+-+-+-+-+-+-+-+-+ +-+-+-+-+


================================================
🆔 | User ID: 186
📍 | User IP: 192.168.29.201
🌈 | Curr Color: 🔴(RED)
👥 | # of users: 3
🔁 | Unidirectional Ring: 186 -> 968 -> 515 -> 186
👉 | My Next ID: 968
================================================

📞 | Listening started... on port 8000

Sent Token msg to user: 968
Passed Token msg ('TOKEN|515') to user: 968
Passed Token msg ('TOKEN|968') to user: 968
👑 | Leader found!
968 is our leader.
```

- **Input File generator for Distributed Mutual Exclusion algorithm :**
  - Generates an input file that contains the detail about the number of users, their IP and ID, ring order, and the need to execute the critical section in a given round of token passes.

- **Distributed Mutual Exclusion Algorithm :**
  - Python Script that implements Distributed Mutual Exclusion (Ring Based) Algorithm on a unidirectional ring.
  - The program reads input.txt and starts simulating the mutual exclusion needed to execute critical sections among all processes in the network.

```
+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+
|M|u|t|u|a|l| |E|x|c|l|u|s|i|o|n|
+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+-+


================================================
🆔 | User ID: 2
📍 | User IP: 192.168.29.247
👥 | # of users: 3
🔁 | Unidirectional Ring: 2 -> 3 -> 1 -> 2
👉 | My Next ID: 3
🔑 | Has token: No
================================================

📞 | Listening started... on port 8000

Recieved TOKEN from 1.
I have TOKEN.
I DO NOT need to enter CS.
Passed TOKEN to user: 3.
Round 0 over.
------------------------------------------------
Recieved TOKEN from 1.
I have TOKEN.
I DO NOT need to enter CS.
Passed TOKEN to user: 3.
Round 1 over.
------------------------------------------------
```

```
Recieved TOKEN from 1.
I have TOKEN.
I NEED to enter CS.
        🏃 Executing CS...
        ✅ Done with CS.
Passed TOKEN to user: 3.
Round 2 over.
------------------------------------------------
Recieved TOKEN from 1.
I have TOKEN.
I NEED to enter CS.
        🏃 Executing CS...
        ✅ Done with CS.
Passed TOKEN to user: 3.
Round 3 over.
------------------------------------------------
Recieved TOKEN from 1.
I have TOKEN.
I NEED to enter CS.
        🏃 Executing CS...
        ✅ Done with CS.
Passed TOKEN to user: 3.
Round 4 over.
```

Challenges

- Setting up the chat app on different Operating systems was difficult, each OS has its own way of providing an IP address. Especially in Linux, while programmatically fetching IP addresses, we got IP of localhost, which was later fixed.
- Debugging distributed applications requires a lot of time. Sometimes there was a deadlock-like stage in the chat app
- Due to all members not being present at the same place, this made it difficult to test and debug.

Contribution

1. **Abhijeet Sonkar:** Coding leader election, Test File generation, testing and debugging, and Report preparation
2. **Avneesh Kumar:** Coding Chat app and leader election and mutual exclusion algorithms, Testing & debugging, and Documentation preparation
3. **Pradhuman Singh Baid:** Coding Chat App and Mutual exclusion algorithm, Testing & debugging, and documentation
4. **Sandeep Sahu:** Coding Input File generators, Interface improvement, Testing & debugging, and report preparation
5. **Shyam Tayal:** Coding Chat App and mutual exclusion algorithm, Testing & Debugging, and report preparation