Project 2

Professor: Md Hasanuzzaman(Zaman) Noor (mdhasanuzzaman.noor@uta.edu)

GTA: Md Samiul Arshad, ([mohammadsamiul.arshad@mavs.uta.edu](mailto:mohammadsamiul.arshad@mavs.uta.edu))

Course: [2222-CSE-5306-001-DISTRIBUTED SYSTEMS](https://uta.instructure.com/courses/101458)

Project2 Contributors:

Part1(Berkley algo) by Azharuddin

Part2(implementation of vector clock and synchronizing lamport clock to vector clock) by Sakshi

Name: Sakshi

**Maverick ID : 1001993702**

Email: sx3702@mavs.uta.edu

Contact No.: +1 3153851203

Course Name: Computer Sci-MS Non Thesis (1st SEM)

Name: Azharuddin Irfani Shaik Mohammed

**Maverick ID : 1002041894**

Email: axs1895@mavs.uta.edu

Contact No.: +1 8178299403

Course Name: Computer Sci-MS Non Thesis (1st SEM)

I have neither given nor received unauthorized assistance on this work

Signed: Azharuddin Irfani Shaik Mohammed

Date: 31-March-2022

# Technical Details

Language: Python 3.8.10

IDE: VS Code

OS: Windows10

time daemon port: 18888

node1 port:18081

node2 port:18082

node3 port:18083

# Supported Operations

## Part 1:

* Clock daemon node accepts connection via socket on a predefined port and starts listening to incoming connections
* Client nodes connect to the Clock daemon on the port
* Each Client sends their respective logical clocks (random number in this case)
* Clock daemon, calculates average and sends back the skew to each node
* Client nodes adjust their clocks as per the skew

## Part 2:

* Node class initializes a node on the given socket provided and starts listening for incoming messages. It takes care of modifying the vector clock when a receive or send event occurs
* Each node object can then be used to send messages to another node.
* At every event happening, ***clock[index]*** ***is increased by 1***
* Whenever a data is received in any node ***max (current, received)*** is calculated and updated.

Steps involved:

* Run ***sh start.sh*** in terminal (kills the existing processes holding port numbers and initiates 3 separate nodes)
* 3 log files will be generated (node1.log, node2.log, node3.log)

Diagram

Description automatically generated

# Testing Steps

**Run python3 time\_daemon.py**

Text

Description automatically generated

**Run sh start.sh in another terminal**

**Start.sh has below code, to kill active port and start node1, node2,node3 python file which will produce 3 log files (node1.log,node2.log,node3.log)**

**Graphical user interface

Description automatically generated with medium confidence**

Node1.log

**Text

Description automatically generated**

Node2.log

Text

Description automatically generated

Node3.log

Text

Description automatically generated

Whenever skew value is 0.0, vector clock process initiates

# Issues Encountered and Learnings

## 1)Handling abrupt disconnection of clients

Whenever a client is disconnected, the server needs to handle the failures else the server crashes.

## 2)Avoid repetitive code while creating nodes

Since we had to demonstrate N node cluster, to improve code reusability we created a Node class to initialize a node with a few lines of code.

## 3)Modifying the vector clock from different threads

Noticed issues when modifying the vector clock without any locking mechanisms in place. Two threads could overwrite the vector clock and we can potentially miss an event. To avoid this, we used a blocking queue to store the vector clock in a thread safe manner. The read and write operations can only be performed by one thread at a time.

## 4)Passing vector clock alongside messages

Used a dictionary/map to store the message body and vector clock. To decode the map on the receiving side, used ast to transform the string representation back to a dictionary. Here I learned more about message serialization and deserialization techniques in Python.

# Appendix

1. <https://docs.python.org/3/library/threading.html>
2. <https://docs.python.org/3/tutorial/datastructures.html>
3. [https://www.geeksforgeeks.org/vector-clocks-in-distributed-systems](https://www.geeksforgeeks.org/vector-clocks-in-distributed-systems/)