Project Report

Title

Develop an n-node distributed system that implements a Vector clock

Parts

This project consists of one part:

- Part-1: Implementing distributed system which uses a vector clock and on every message received/sent from/to other system prints before and after clock values

Part-1

In this part, there are three scripts, which are:

1. generic.py
2. node.py
3. sockets.py

generic.py

- This file's primary purpose is to take the total number of systems (i.e., NODE_COUNT) from the user and then find many available ports to create socket servers. By this, we are mimicking multiple systems of distributed systems in a single machine.
- After finding ports for different systems, Program will automatically spawn separate systems in console mode.

node.py

- This file is a child system script that will govern how our systems will behave on messages received or sent. This script will be executed by generic.py, But we can also manually execute it by providing the required command-line arguments.
- It will divide the whole required functionality of the child system into two parts:
    - Sender
    - Receiver
- To accommodate these two functions, we have used the multithreading concept. Here sender is running on an exitable infinite for loop in the main thread while the Receiver is running in a separate thread.
- The primary purpose of the **sender** is to take "message" and "systems id to send message to" from the user as an input and then send the message to another specified system.
- The Receiver's primary purpose is to create a server using sockets that listens to a specified port for any incoming message.

sockets.py

- This script file consists of different helper utility functions for the node.py file. It has functions to create a server for a specific port, handle vector clock

changes for the system and send requests to a specific port/system with a message. Another helper function to print a Help message and a dummy internal process for the system is also included in this file.

Learnings

- How to use sockets to communicate with other systems.
- How Vector Clock in the distributed system works and how to implement it.

Difficulties encountered

We initially used the steps below to determine whether the port is available.

1. Get any port number
2. Try to connect with that port (as a sender)
3. If we can connect with it, then it is available to use
4. Otherwise, choose the next port and repeat steps 1-4

But, this method was time-consuming in that it took 2-3 seconds on each cycle from steps 1-4. It was not an optimized solution.

To optimize it then, we used the below steps.

1. Get any port number
2. Try to bind the socket with that port (as a listener/server)
3. If we can connect with it, then it is available to use
4. Otherwise, choose the next port and repeat steps 1-4

It reduced the finding time of available ports from 2-3 seconds per port to little time for the whole list!

Conclusion

The whole project was tested and implemented as defined in the project statement. Code for the project is in the source_code directory and distributed in different files as mentioned above. README file is in the same project directory.