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 vector clock and on every message received/send 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

Main purpose of this file is to take total number of systems (i.e., NODE\_COUNT) from user and then finding that many available ports to create socket servers. By this we are mimicing multiple systems of distributed systems in single machine.

After finding ports for different systems, it will autometically spawn separate systems in console mode.

### node.py

This is child system script which will govern how our systems will behave on message receive or send. This script will be executed by generic.py, But we can also manually execute it by providing required command line arguments.

This will divide whole required functionality of child system in two parts:

* Sender
* Receiver

To accommodate these tow functionality we have used multithreading concept. Here sender is running on exitable infinite for loop in main thread while receiver is running in saperate thread.

Main purpose of **sender** is to take “message” and “systems id to send message to” from user as an input and then send message to other specified system.

Main purpose of **Receiver** is to create server using sockets that listens on specified port for any incoming message.

### sockets.py

This script file consists different helper utility function for node.py file. It have functions to create server for specific port, handle vector clock changes for system and send request to specific port/system with message. Other helper function to print Help message and dummy internal process for system as also included in this file only.

## Learnings

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

## Dificulties encountered

We were initially using below steps to find out if port is available to use or not.

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 next port and repeat steps 1-4

But, this method was time consuming in a way that it was taking 2-3 seconds on each cycle from steps 1-4. This was clearly not optimized solution.

To optimize it then we used below steps.

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

This reduced port finding time from 2-3 seconds per port to negligible time for whole list!

## Conclusion

Whole project was tested and implemented as defined in the project statement. Code for project is written in one directory in different files as mentioned above. README file can be found in the same directory.