Vector Clock Process

by

Divya Boggavarapu(1002086719)

Abhishek Reddy Yaramala(1002072996)

I have neither given nor received unauthorised assistance on this work.

Sign: Divya Boggavarapu Date: 10/08/2022

Sign: Abhishek Reddy Yaramala Date: 10/08/2022

Table of Contents

Project Description:

This program is to develop an n-node distributed system that implements a vector clock. The distributed system uses a logical clock to timestamp messages sent/received among the nodes. We used java programming language. The distributed system will be emulated using multiple processes on a single machine. Each process represents a machine and has a unique port number for communication.

Implemented the vector clock for your distributed system. We created two threads for each process, one for sending messages to other nodes and one for listening to its communication port. Communication among nodes was done using sockets. Once a process sends a message, it will print its vector clock before and after sending a message. Similarly, once a process receives a message, it will print its vector clock before and after receiving the message. We did it for 3 processes which will not fail, join, or leave the distributed system.

Contribution:

This project is developed by Divya Boggavarapu(1002086719) and Abhishek Reddy Yaramala(1002072996).

Requirements

- Java
- Eclipse IDE

Java Socket

A socket is an endpoint for communication between two machines. The actual work of the socket is performed by an instance of the SocketImpl class. An application, by changing the socket factory that creates the socket implementation, can configure itself to create sockets appropriate to the local firewall.

java.net.Socket class.

Socket socket = new Socket(server, port);

Vector Clocks in Distributed Systems

Vector Clock is an algorithm that generates partial ordering of events and detects causality violations in a distributed system.
These clocks extend Scalar time and determine if a contributing event has contributed to another event in the distributed system in order to enable a causally coherent representation of the distributed system. In essence, it encompasses all the causal connections. This approach enables us to mark each process in the system with a vector (a list of numbers) containing an integer for each local clock. So there will be a vector or array of size N for each of the N processes specified.

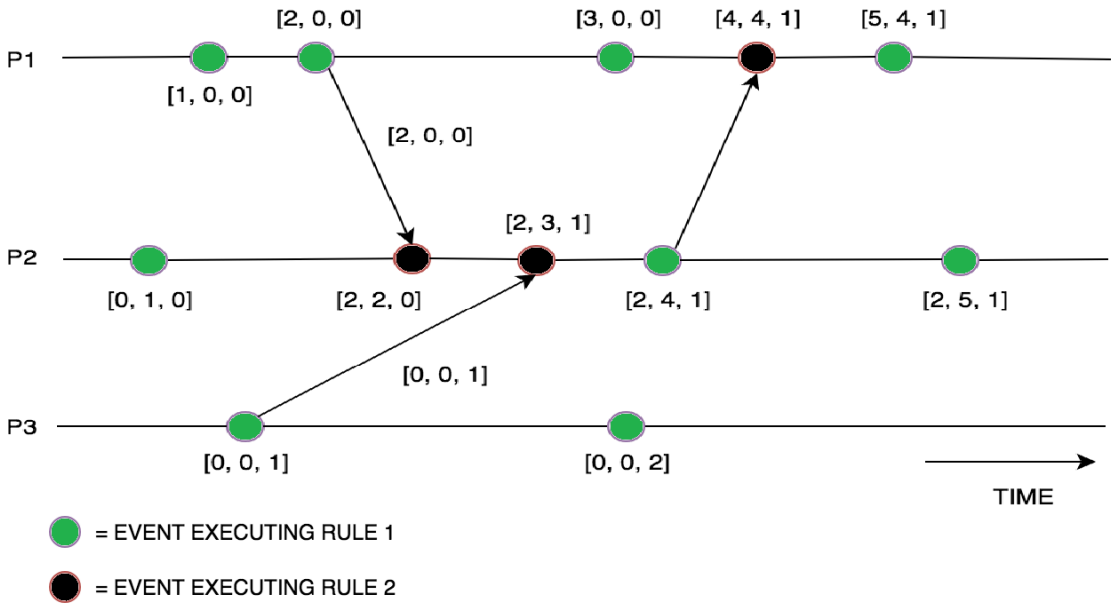
What is the vector clock algorithm's operation?

- All of the clocks are initially set to zero.
- The logical clock value for each process in the vector is increased by one whenever an internal event happens in that process.
- Additionally, each time a process sends a message, its logical clock's value in the vector is increased by one.

Every time a process receives a message, its logical clock's value in the vector is increased by one. In addition, each element is updated by summing the values in the received message's vector and the process's own vector clock (for every element).

Example:

Consider a process (P) where each process has a vector of size N. The vector clock is responsible for carrying out the aforementioned set of rules:



The arrows show how the values of the vectors are communicated between the processes in the example above, which shows the technique by which the vector clocks are updated after the completion of internal events (P1, P2, P3).

In summary, vector clock methods are employed in distributed systems to give a causally consistent ordering of events, but the vector clocks must be kept in sync by sending the complete vector to each process whenever a message is sent.

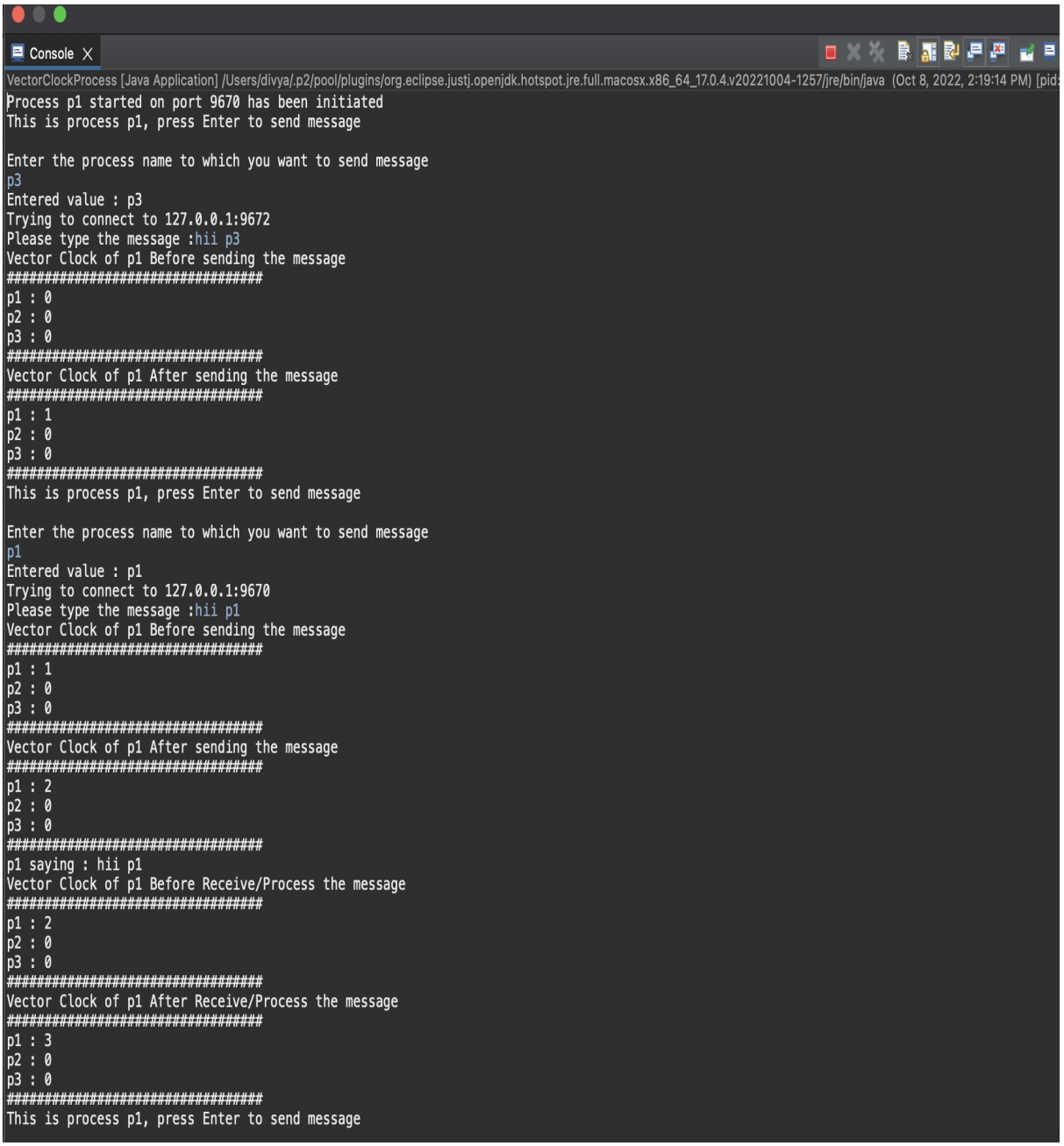
Implementation Steps:

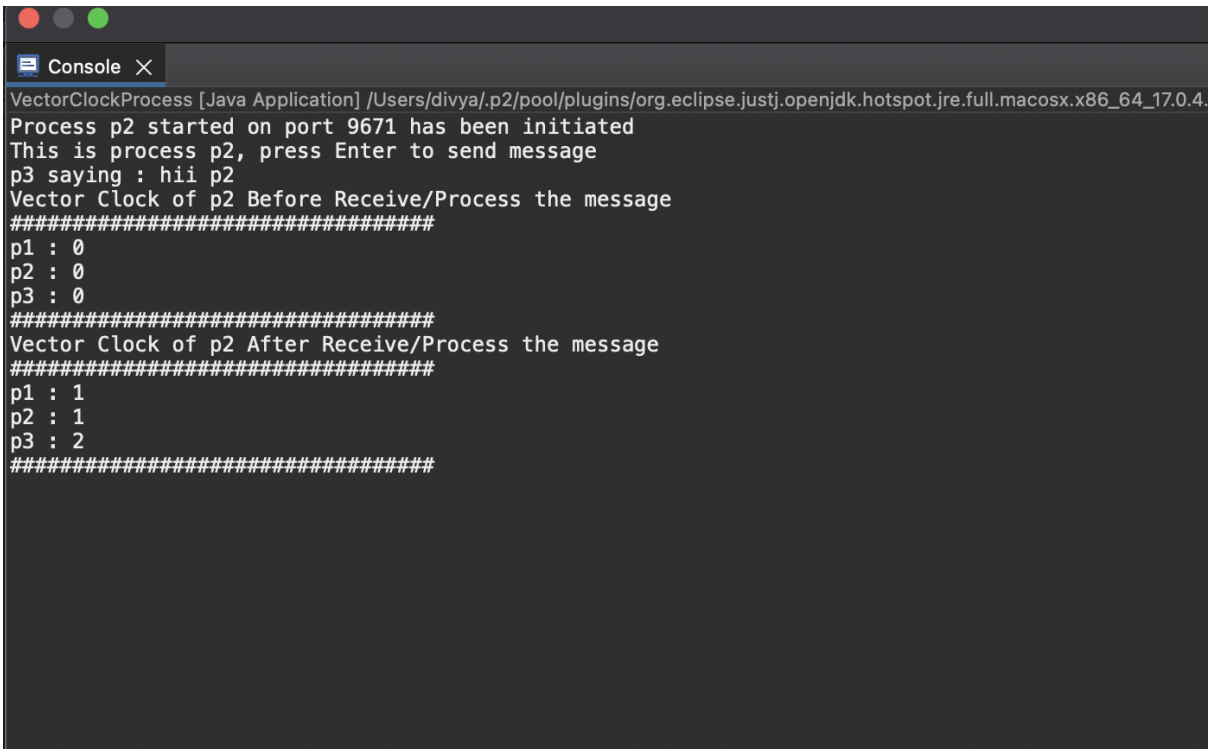
- Created a VectorClockProcess.Java where we initialised the processes, Ports and vector clock.
- Created a Hashmap for our processes and initialised them.
- Initialised the socket and created two threads, one for sending messages and another for receiving the messages
- In SenderThread.Java we established the connection through socket between our processes and sending the message to the other processes and printing the vector clock upon sending the message.
- In ReaderThread.Java we are reading the message from SenderThread.Java and updating the vector clock system accordingly.

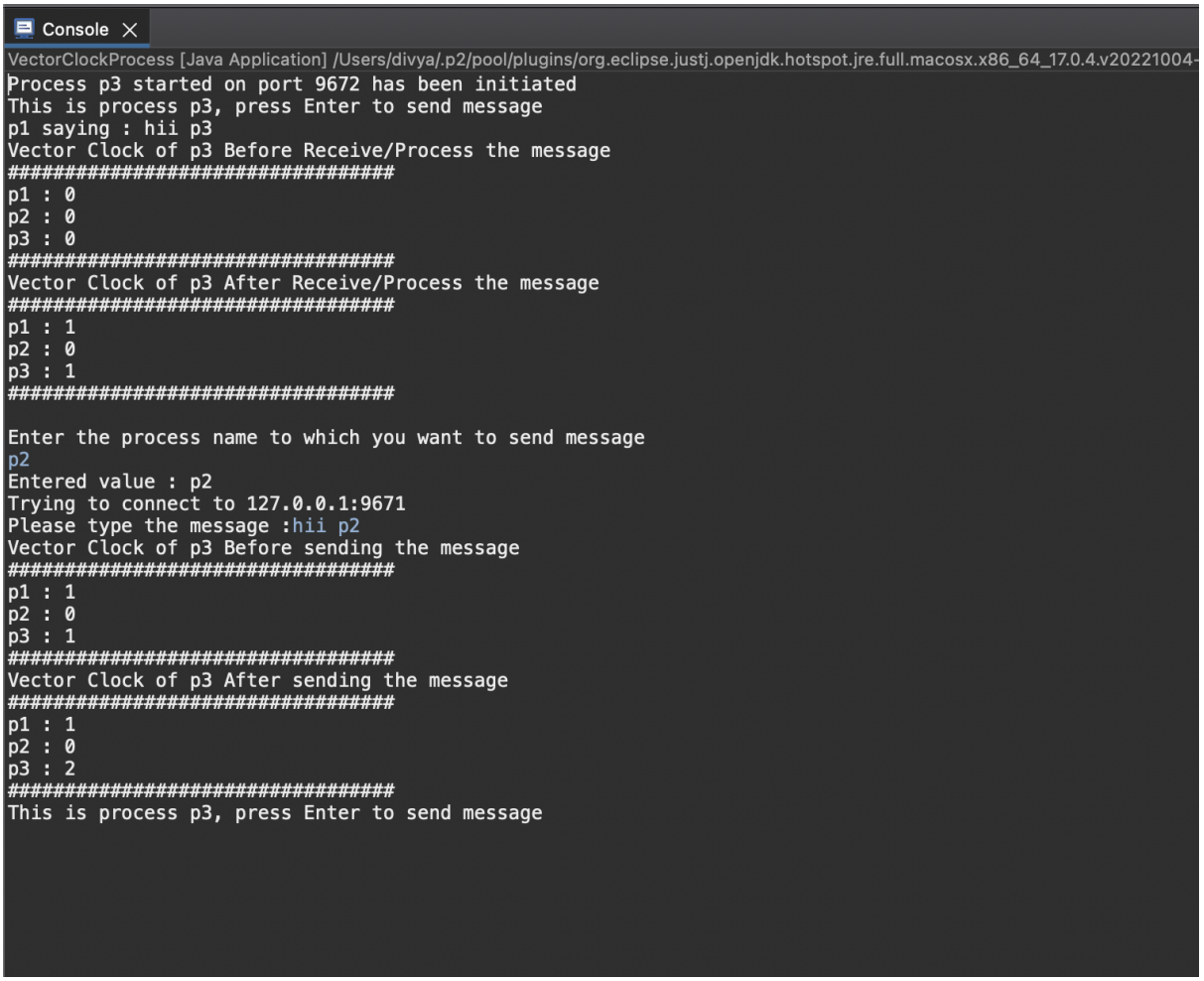
Detailed Instructions to Execute Code

- Right click on the VectorClockProcess.java and run as 1 Java application for 3 times.
- Three java processes will start running
- It will give the message 'Process p1 started on port 9670 has been initiated, This is process p1, press Enter to send message'
- Once we click enter, Please type Process name(p1, p2, p3) to which you want to send message and message text
- It'll display the vector clock before sending the message and after sending the message
- change the console and repeat the same procedure for all the processes by changing the console, we can see the updated vector clock.

Screenshots:




Console
VectorClockProcess [Java Application] /Users/divya/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_17.0.4.
Process p2 started on port 9671 has been initiated
This is process p2, press Enter to send message
p3 saying : hii p2
Vector Clock of p2 Before Receive/Process the message
##################################
p1 : 0
p2 : 0
p3 : 0
##################################
Vector Clock of p2 After Receive/Process the message
##################################
p1 : 1
p2 : 1
p3 : 2
##################################


Console
VectorClockProcess [Java Application] /Users/divya/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_17.0.4.v20221004-
Process p3 started on port 9672 has been initiated
This is process p3, press Enter to send message
p1 saying : hii p3
Vector Clock of p3 Before Receive/Process the message
##################################
p1 : 0
p2 : 0
p3 : 0
##################################
Vector Clock of p3 After Receive/Process the message
##################################
p1 : 1
p2 : 0
p3 : 1
##################################
Enter the process name to which you want to send message
p2
Entered value : p2
Trying to connect to 127.0.0.1:9671
Please type the message :hii p2
Vector Clock of p3 Before sending the message
##################################
p1 : 1
p2 : 0
p3 : 1
##################################
Vector Clock of p3 After sending the message
##################################
p1 : 1
p2 : 0
p3 : 2
##################################
This is process p3, press Enter to send message

What issues encountered

- Took some time for logic to update the vector clock according to processes.

What we learnt

- Got to learn about vector clock system in distributed systems.