

# PROJECT 1 REPORT

**Yashwanth Rao Varala Balaji** – 1001916698

**Rakshith Singh Thakur** – 1002037934

I have neither given nor received unauthorized assistance on this work. I will not post the project description and the solution online.

Sign: Yashwanth Rao Varala Balaji, Rakshith Singh Thakur Date: 02/13/2023

Objective:

The main objective of this project is to develop a distributed system comprising n nodes, with each node representing a machine and communicating with other nodes through message passing. To timestamp the messages exchanged between nodes, the system utilizes a vector clock.

For ease of design and testing, the distributed system will be simulated using multiple processes on a single machine, and each process will have a unique port number to enable communication between the nodes.

This project is not restricted to a specific programming language, allowing for flexibility in implementation. The aim is to establish a functional distributed system that can efficiently exchange messages and maintain a consistent vector clock across all nodes.

Part-1: Unicast Messaging

Unicast messaging means that each process can send a message to another process. When a message is sent and received, the vector clocks of the processes involved are updated, and their values are printed before and after sending/receiving the messages.

To implement unicast messaging, a process can start a message to another process by providing the ID of the destination process and the message content. Before sending the message, the vector clock of the sending process is increased. The message is then sent to the destination process, and the receiving process updates its vector clock using the received vector clock. The vector clocks of the sending and receiving processes are printed before and after sending/receiving the message.

Part-2: Broadcast Messaging

Broadcast messaging means that each process can send a message to all other processes in the distributed system. When the message is sent and received, the vector clocks of all processes are updated, and their values are printed before and after sending/receiving the messages.

To implement broadcast messaging, a process can start a broadcast message by providing the message content. Before sending the message, the vector clock of the sending process is increased. The message is then sent to all other processes in the distributed system, and each receiving process updates its vector clock using the received vector clock. The vector clocks of all processes are printed before and after sending/receiving the message.

Results:

The unicast and broadcast messaging functionalities were successfully implemented in the distributed system, and all processes were able to communicate with each other. The vector clocks were updated accurately, and their values were printed before and after sending/receiving messages. The system was able to maintain a consistent vector clock across all nodes, ensuring that messages were properly ordered and synchronized. Overall, the implementation was successful in achieving the project goals of developing a functional distributed system that implements a vector clock for timestamping messages.

Learning from the project:

1. By implementing the project, we gained better understanding of distributed systems and how they work.
2. Helped us in developing skills in programming languages and technologies used for building distributed systems, such as RPC and socket programming.
3. Understood logical clocks and their use in distributed systems for timestamping and ordering events.

Issues faced:

During implementation of the project we faced issues trying to find the resources since this was our first time working with sockets. Updating the vector clocks in different machines was not working.