Simulation of Multicast Message Delivery

Jahnavi Redrouthu
Computer Science
University of Missouri-Kansas City
Kansas City, MO, USA
jrdhp@umsystem.edu

Rupasree Guttikonda
Computer Science
University of Missouri-Kansas City
Kansas City, MO, USA
rg93m@umsystem.edu

Sathwika Cherukuri
Computer Science
University of Missouri-Kansas City
Kansas City, MO, USA
scf7p@umsystem.edu

ABSTRACT

This project focuses on developing a discrete event simulation to model reliable multicast communication within distributed systems. The simulation ensures ordered and consistent message delivery among nodes, incorporating mechanisms for message ordering, fault tolerance, and recovery. Additionally, a secondary simulation demonstrates basic PING-PONG message exchanges between two nodes. This report outlines the problem statement, reviews related work, details the proposed techniques, and presents preliminary implementation results. The findings show the effectiveness of the simulation framework in addressing core challenges in distributed communication systems.

INTRODUCTION

In distributed systems, efficient and reliable multicast communication is vital for applications requiring consistent data dissemination among multiple nodes. Ensuring that messages are delivered in order and consistently across all recipients poses significant challenges, especially in the presence of network delays and node failures. This project aims to simulate such an environment using discrete event simulation (DES) techniques to study and improve multicast message handling.

This simulation serves as a platform for testing distributed computing concepts like event-driven communication, fault recovery, and consistency. The primary goals are to:

- Simulate multicast communication in a distributed environment.
- Ensure ordered and reliable message delivery.
- Implement fault tolerance and recovery.
- Measure the performance of message exchanges under variable conditions.

RELATED WORK

1. Causal Ordering in Distributed Systems

Causal ordering ensures that if one message causally precedes another, all recipients receive them in that order. Protocols like the ISIS system utilize vector timestamps to maintain this order, ensuring consistency across distributed nodes.

2. Scalability in Reliable Multicasting

As the number of nodes increases, scalability becomes a concern. Techniques such as hierarchical structuring and efficient acknowledgment schemes have been proposed to address the challenges of scalable and reliable multicast communication.

3. Group Communication Mechanisms

Group communication involves managing message delivery among a set of nodes. Various protocols ensure reliable and ordered delivery, addressing challenges like message loss and dynamic group membership.

4. Reliable Multicast Protocols

Protocols such as SRM (Scalable Reliable Multicast) and Pragmatic General Multicast (PGM) have been developed to address specific scalability and reliability challenges in distributed systems. These solutions incorporate mechanisms like negative acknowledgments (NAKs), FEC (forward error correction), and hierarchical node organization.

5. Event-Driven Simulations

Discrete event simulation (DES) is a common approach in evaluating distributed systems. Tools like OMNeT++ and NS2/NS3 provide simulation environments for network protocol behavior, validating theoretical concepts under practical constraints.

PROPOSED TECHNIQUE

1. Discrete Event Simulation Framework

The simulation employs a DES framework to model network events, including message transmissions, receptions, and node failures. An event queue manages these events chronologically, allowing for precise control and observation of the system's behavior.

2. Multicast Message Handling

Each host node maintains a message log with sequence numbers to ensure ordered delivery. Upon receiving a message, nodes check sequence numbers to detect missing messages, prompting retransmission requests when necessary.

3. Fault Tolerance and Recovery

The simulation introduces random node failures to assess fault tolerance mechanisms. Nodes utilize heartbeats to detect failures and employ leader election algorithms to reconfigure the network, ensuring continued operation despite node outages.

4. PING-PONG Messaging

A secondary simulation models two nodes exchanging PING and PONG messages based on scheduled events. This setup aids in analyzing basic message exchange dynamics and latency measurements.

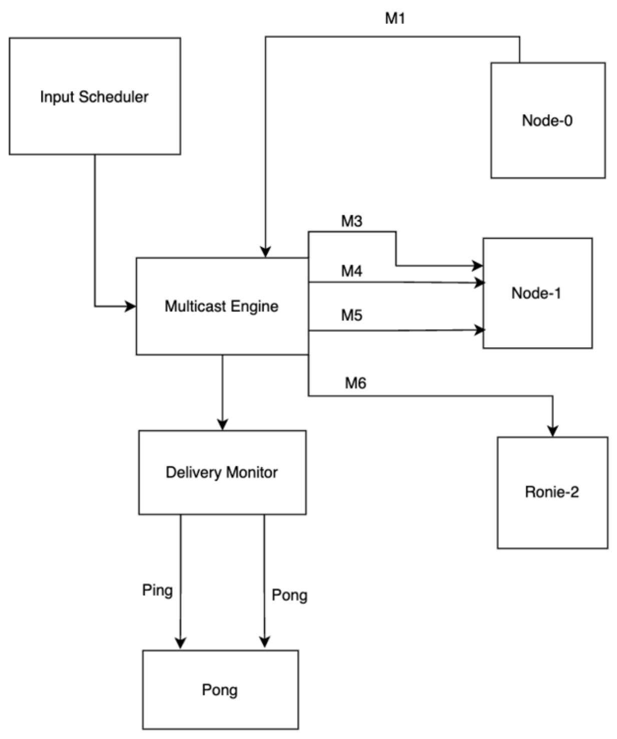


Figure 1: Simulation of multi message delivery

In the firstorder.py simulation, multiple nodes (like Node-1 and Node-2) act like computers in a network, sending multicast messages (e.g., M1, M4) to all other nodes. These messages are labeled (like ”janu”, “rupa” or “sathwi”) to represent their content. The simulation shows how nodes in distributed systems send and coordinate messages over time — just like in real-world networks.

Node-0 receives multicast messages from other nodes but only delivers them after receiving its own forwarded copy. This ensures reliable, ordered, and duplicate-free delivery — essential for consistency in distributed systems like chat apps or trading platforms.

The PING-PONG mechanism in distributed systems is used to check if nodes are alive and connected by exchanging messages. Delays between messages simulate real network latency. The simulation ends when all such interactions are completed

5. Distributed Computing Aspects

The project incorporates several distributed computing paradigms:

- Group Communication: Manages message dissemination among multiple nodes.
- Fault Detection and Recovery: Implements mechanisms to detect and recover from node failures.
- Event Scheduling: Utilizes an event-driven approach to simulate asynchronous operations.
- Client-Server and Peer-to-Peer Patterns: Simulated nodes exhibit both centralized (leader election) and decentralized (peer) behaviors.

PRELIMINARY RESULTS AND IMPLEMNTATION

1. Simulation Environment

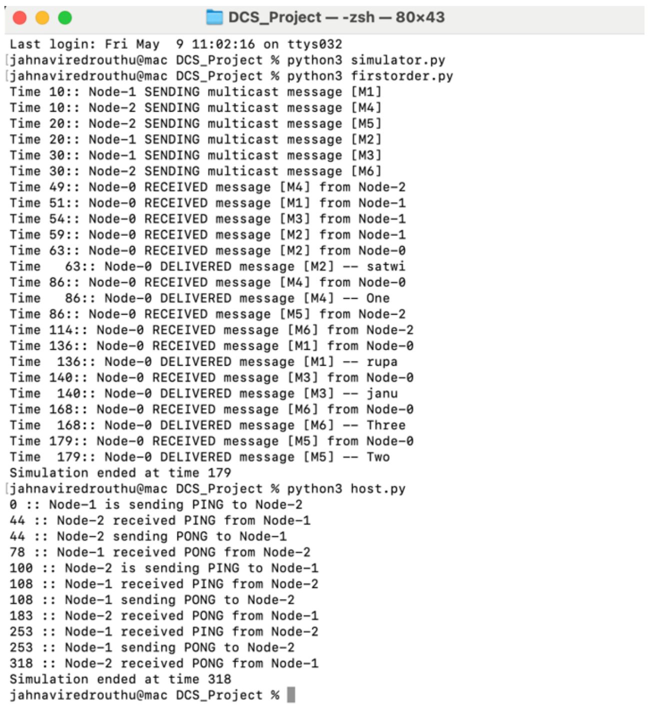
The simulation is developed in Python, leveraging object-oriented design for modularity. The DES framework models network behaviour, including message delays and node failures. Nodes, messages, and events are abstracted into classes with specific behavior definitions.

2. Initial Findings

Preliminary tests demonstrate that the simulation successfully maintains message order and consistency under normal conditions. Introducing random delays and node failures allowed for evaluating the effectiveness of the implemented fault detection and recovery mechanisms.

3. Challenges Encountered

- **Synchronization Issues:** Ensuring all nodes have a consistent view of the message order required implementing synchronization mechanisms.
- **Performance Bottlenecks:** Simulating large networks introduced performance challenges, necessitating optimization of the event handling system.
- **Network Delay Simulation**: Variable delay models helped in evaluating robustness against real-world conditions.



DCS_Project — -zsh — 80×43

```
Last login: Fri May  9 11:02:16 on ttys032
jahnaviredrouthu@mac DCS_Project % python3 simulator.py
jahnaviredrouthu@mac DCS_Project % python3 firstorder.py
Time 10:: Node-1 SENDING multicast message [M1]
Time 10:: Node-2 SENDING multicast message [M4]
Time 20:: Node-2 SENDING multicast message [M5]
Time 20:: Node-1 SENDING multicast message [M2]
Time 30:: Node-1 SENDING multicast message [M3]
Time 30:: Node-2 SENDING multicast message [M6]
Time 49:: Node-0 RECEIVED message [M4] from Node-2
Time 51:: Node-0 RECEIVED message [M1] from Node-1
Time 54:: Node-0 RECEIVED message [M3] from Node-1
Time 59:: Node-0 RECEIVED message [M2] from Node-1
Time 63:: Node-0 RECEIVED message [M2] from Node-0
Time   63:: Node-0 DELIVERED message [M2] -- satwi
Time 86:: Node-0 RECEIVED message [M4] from Node-0
Time   86:: Node-0 DELIVERED message [M4] -- One
Time 86:: Node-0 RECEIVED message [M5] from Node-2
Time 114:: Node-0 RECEIVED message [M6] from Node-2
Time 136:: Node-0 RECEIVED message [M1] from Node-0
Time  136:: Node-0 DELIVERED message [M1] -- rupa
Time 140:: Node-0 RECEIVED message [M3] from Node-0
Time  140:: Node-0 DELIVERED message [M3] -- janu
Time 168:: Node-0 RECEIVED message [M6] from Node-0
Time  168:: Node-0 DELIVERED message [M6] -- Three
Time 179:: Node-0 RECEIVED message [M5] from Node-0
Time  179:: Node-0 DELIVERED message [M5] -- Two
Simulation ended at time 179
jahnaviredrouthu@mac DCS_Project % python3 host.py
0 :: Node-1 is sending PING to Node-2
44 :: Node-2 received PING from Node-1
44 :: Node-2 sending PONG to Node-1
78 :: Node-1 received PONG from Node-2
100 :: Node-2 is sending PING to Node-1
108 :: Node-1 received PING from Node-2
108 :: Node-1 sending PONG to Node-2
183 :: Node-2 received PONG from Node-1
253 :: Node-1 received PING from Node-2
253 :: Node-1 sending PONG to Node-2
318 :: Node-2 received PONG from Node-1
Simulation ended at time 318
jahnaviredrouthu@mac DCS_Project % 
```

Figure 2: Output Snippet

CONCLUSION

This project successfully demonstrates the application of discrete event simulation to model reliable multicast message delivery in distributed systems. Through systematic event handling, fault detection, and recovery mechanisms, the simulation reflects the complexities of real-world distributed environments. Future work can include scaling to larger node sets, integrating Byzantine fault tolerance, and real-time visual dashboards.

AUTHOR CONTRIBUTIONS

This project was carried out collaboratively by a team of three members, with tasks distributed based on individual strengths and interest areas to ensure a balanced and efficient workflow.

Jahnavi Redrouthu led the design and development of the Discrete Event Simulation (DES) framework, including the event scheduler and base classes for messages and nodes. She was also responsible for integrating message ordering mechanisms using sequence numbers and ensuring the correctness of multicast delivery logic.

Sathwika Cherukuri focused on implementing the multicast message delivery subsystem, including the tracking of acknowledgments and random delay simulations. She also conducted fault injection experiments and implemented the fault detection and recovery strategies using timeouts and heartbeat mechanisms.

Rupasree Guttikonda worked on building the two-node PING-PONG message exchange module and helped analyse the preliminary simulation results. She also handled the configuration of test scenarios and contributed to writing the Related Work and Preliminary Results sections of the report.

REFERENCES

1. Tanenbaum, A. S., & Wetherall, D. J. (2011). Computer Networks (5th ed.). Pearson.
2. Birman, K. P., & Joseph, T. A. (1987). Reliable Communication in the Presence of Failures. ACM Transactions on Computer Systems, 5(1), 47-76.
3. Lamport, L. (1978). Time, Clocks, and the Ordering of Events in a Distributed System. Communications of the ACM, 21(7), 558-565.
4. Chockler, G., Keidar, I., & Vitenberg, R. (2001). Group Communication Specifications: A Comprehensive Study. ACM Computing Surveys, 33(4), 427-469.
5. Omnet++, A. (2021). Introduction to Discrete Event Simulation. Omnet++ Documentation.
6. Floyd, S., Jacobson, V., Liu, C. G., McCanne, S., & Zhang, L. (1997). A Reliable Multicast Framework for Light-weight Sessions and Application-Level Framing. IEEE/ACM Transactions on Networking, 5(6), 784-803.

LINKS

Presentation video

Git Hub Link