

Task 4 – Enhancing Availability and Reliability in a Publish–Subscribe Middleware

1. Introduction

The current Pub/Sub middleware uses a single server for all communication, creating a **single point of failure**. Task 4 proposes a distributed architecture to improve **availability, reliability, and scalability**.

2. Problem with the Existing Single-Server Architecture

- **Single Point of Failure**

If the server crashes or becomes unavailable due to hardware failure, software bugs, or network issues, the entire Pub/Sub system stops functioning.

- **Low Availability**

During server downtime, publishers cannot publish messages and subscribers cannot receive updates.

- **Reduced Reliability**

Messages in transit may be lost if the server fails unexpectedly.

These limitations make the system unsuitable for real-world, large-scale, or mission-critical applications.

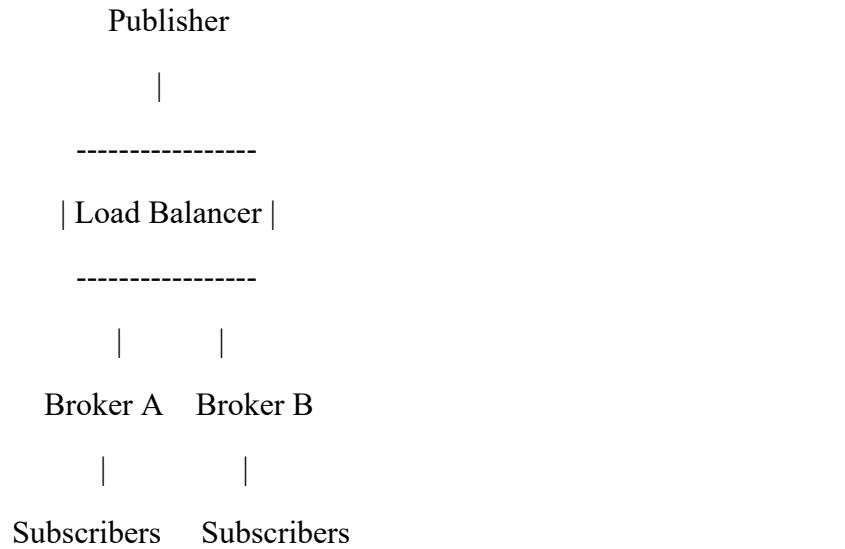
3. Proposed Distributed Pub/Sub Architecture

A **Distributed Broker-Based Publish–Subscribe Architecture** design introduces multiple brokers and a load balancer.

3.1 Components of the Proposed Architecture

- **Load Balancer:** Distributes traffic, handles broker failures.
- **Multiple Brokers:** Handle publishers/subscribers in parallel, support topic-based routing.
- **Replication/Synchronization (optional):** Ensures metadata consistency and reduces message loss.

3.2 Architecture Diagram (Conceptual)



4. Improvements Achieved by the Proposed Architecture

4.1 Improved Availability

- If one broker fails, publishers and subscribers can continue communication through another active broker.
- The system remains operational even during partial failures.

4.2 Improved Reliability

- The risk of message loss is reduced due to multiple brokers.
- Client reconnections can be handled seamlessly by other brokers.

4.3 Improved Scalability

- Additional brokers can be added to handle increased load.

5. Summary

5.1 Comparison

Feature	Single Server	Distributed Architecture
Availability	Low	High
Reliability	Low	High
Scalability	Limited	High
Fault Tolerance	None	Supported

5.2 Conclusion

The single-server-based Publish and Subscribe middleware suffers from a critical single point of failure, limiting its availability and reliability. By introducing a distributed architecture with multiple broker servers and a load balancer, these limitations can be effectively addressed. The proposed architecture ensures continuous service availability, improves reliability by reducing message loss, and enables scalability for future growth. This design makes the Pub/Sub middleware more robust and suitable for real-world distributed systems.