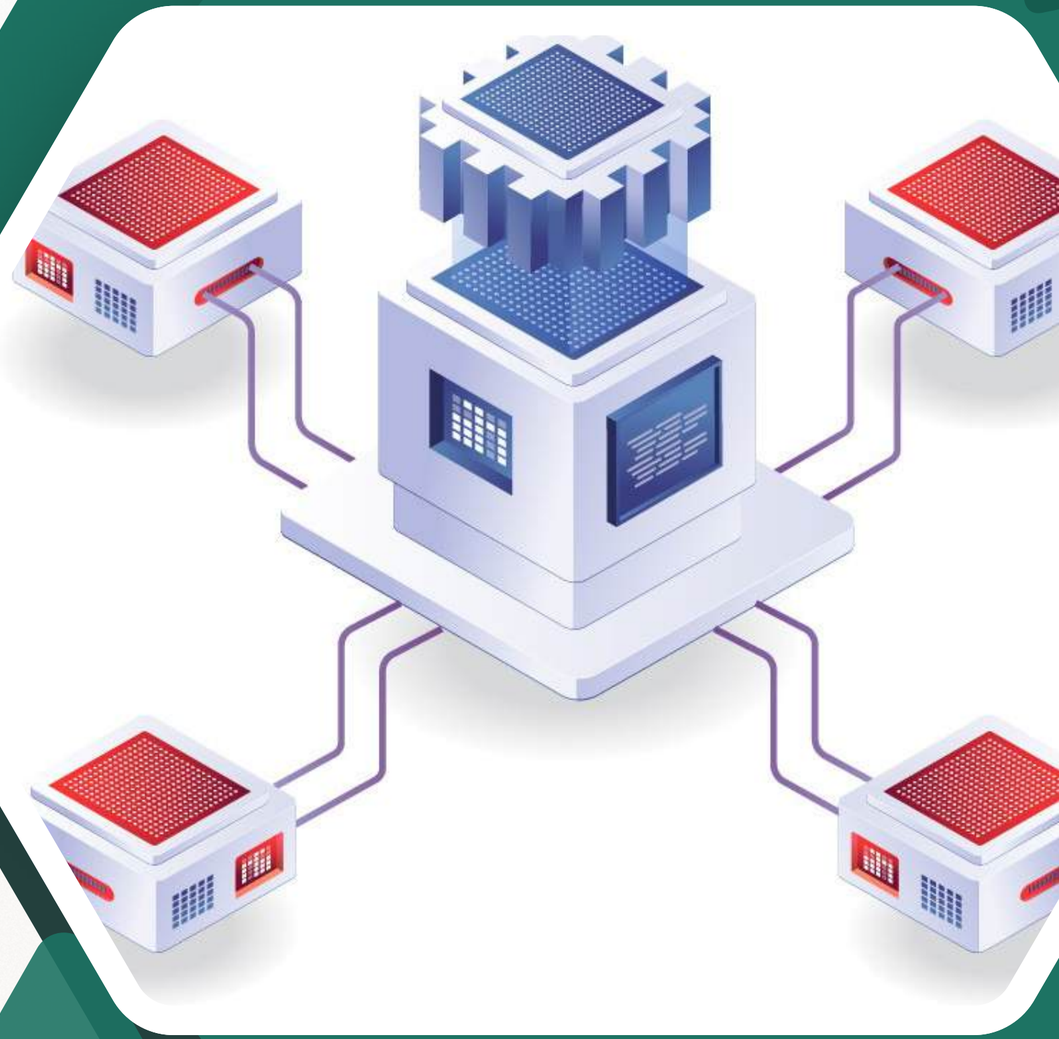
Group 10

DISTRIBUTED MESSAGING SYSTEM



GROUP MEMBERS

- M.A.L.R. KARUNASEKARA - IT23464032
- W.W.G.S.N. GUNAWARDHANA - IT23392820
- H.A.H. SENADHEERA - IT23162082
- M.F.M. AMRI - IT23194144

INTRODUCTION

- Distributed messaging systems are critical infrastructure in modern distributed applications
- Challenges: network partitions, node failures, message ordering, and consistency guarantees
- Need for fault tolerance to ensure continuous operation despite failures

PROJECT OBJECTIVES

- Design and implement a fault-tolerant messaging system using primary-backup architecture
- Incorporate simplified Raft consensus principles for leader election
- Ensure message durability through replication
- Maintain system availability during component failures

SYSTEM OVERVIEW

Two-server configuration (primary on port 12345, backup on port 12346)

Java-based implementation with TCP sockets for communication

Support for multiple concurrent clients

SYSTEM ARCHITECTURE

Core Components

Message Class

- Sender ID (String)
- Recipient ID (String)
- Content (String)
- Timestamp (Date)
- Unique message ID for tracking

RaftNode Class

- States: LEADER, FOLLOWER, CANDIDATE
- Term tracking for consensus
- Vote counting mechanisms
- State transition logic

ClientHandler Class

- Manages individual client connections
- Parses client commands (SEND, RECEIVE, LIST)
- Routes messages to appropriate handlers

Server Class

- Socket initialization and connection acceptance
- Thread management for client handlers
- Message queue maintenance
- Replication coordination

IMPLEMENTATION

Primary-Backup Replication

Replication Protocol:

- TCP socket connection to backup server
- Java object serialization for message transmission
- Synchronous wait for acknowledgment
- Timeout handling for backup unavailability

Message Delivery:

- Messages stored in recipient-specific queues
- Delivered upon client request (pull model)
- Timestamp-based ordering within queues

IMPLEMENTATION

Fault Tolerance Mechanisms

Message Durability:

- Messages persisted on both primary and backup
- Acknowledgment only after successful replication
- In-memory queues with potential for disk persistence

Server Failure Handling:

- Manual promotion of backup
- Backup failure: Primary continues operation with degraded durability
- Client reconnection logic with server discovery

IMPLEMENTATION

Time Synchronization Mechanisms

Clock Management:

- Each server maintains a hybrid logical-physical clock
- Periodic synchronization using a simplified NTP-like protocol
- Gradual clock adjustment to avoid abrupt time changes

Message Ordering:

- Messages timestamped on arrival using local clock
- Priority queues used to deliver messages in correct order
- Small delay window to reorder out-of-sequence messages

Skew Handling:

- Clock drift detected via periodic reference sampling
- Automatic correction when deviation exceeds threshold
- Trade-off: Slightly increased latency for improved ordering accuracy

IMPLEMENTATION

Consensus Mechanisms

Consensus Algorithm:

- Raft protocol used for agreement on message order and leadership
- Three node states: Leader, Follower, Candidate
- Term-based leader election with majority voting

Log Replication:

- Append-only log ensures consistent message storage across nodes
- New messages committed after majority acknowledgment
- Recovery protocol for nodes falling behind

Failure Handling:

- Heartbeats detect leader/node failures
- Automatic leader re-election on failure
- Guarantees: Consistency prioritized, availability may be reduced during leader transitions

EVALUATION AND LIMITATIONS

Successful Aspects

Replication Performance:

- Low latency for local deployment (<10ms per message)
- Throughput of ~1000 messages/second in testing

Fault Tolerance:

- No message loss when backup is available
- System continues operation with manual intervention after primary failure

Consistency Guarantees:

- Strong consistency for replicated messages
- Timestamp-based ordering within single recipient queues

Current Limitations

Failure Detection:

- No automated heartbeat mechanism
- Relies on connection exceptions to detect failures
- Manual intervention required for recovery

Scalability Constraints:

- Two-node configuration only
- No support for dynamic cluster membership
- In-memory message storage limits capacity

Consistency Challenges:

- No global message ordering across recipients
- Clock synchronization issues affect timestamp reliability
- No persistent log for crash recovery

THANK YOU