Name: Soham Sonar

CWID: A20541266

Table of Contents:

1. Overview

This design document describes the enhancements to the decentralized Peer-to-Peer Publisher-Subscriber System. Building on the PA3 design, the system now supports **replicated topics** for fault tolerance and performance optimization, and **dynamic topology configuration** to accommodate runtime changes in network nodes.

---

2. Objectives

- **Performance Optimization:** Reduce latency by replicating topics closer to clients.
- **Fault Tolerance:** Ensure continuous system operation despite node failures.
- **Dynamic Adaptability:** Allow seamless addition and removal of nodes in the hypercube topology.
- **Data Consistency:** Maintain consistency of replicated topics across nodes.

---

3. System Architecture

The enhanced system retains the **hypercube topology** and **Distributed Hash Table (DHT)** but incorporates new features to address performance and fault tolerance:

1] Replicated Topics for performance optimization & fault Tolerance.

2] Dynamic Topology configuration (add and removal of hypercube nodes)

node [000] : Neighbours : [001, 010, 100]

(Data)

Replica

~~steps~~: 1] First implement replicas of data in a node to its neighbours.

create topic ~~from~~ in 000

↓

Def Create topic:

locally saved

↳ send request (replica function - connect (001), (010), (100))

↳ [001, 010, 100]

↳ enter in 001 replica function

Def Replica:

Save topic locally in self.replica

↳ enter 010

. . . .

↳ enter 100

. . . .

operation complete.

- **Replication Layer:** Extends the DHT with replication capabilities, enabling topics to exist on multiple nodes.
- **Fault Detector:** Detects node failures and reroutes requests to nodes with replicated topics.

---

4. Key Features

4.1 Replicated Topics

1. **Performance Optimization:**
   - Topics are replicated on nodes physically closer to clients for reduced access latency.
   - Replica placement decisions consider factors like network latency, client access frequency, and node load.
2. **Fault Tolerance:**
   - Replicas ensure continuity in topic availability during node failures.
   - A synchronization mechanism ensures consistency across all replicas.
3. **Consistency Model:**
   - Adopt a **Primary-Secondary Replication Model**:
     - **Primary Node**: Handles all write operations.
     - **Secondary Nodes**: Handle read operations and synchronize periodically.

4.2 Dynamic Topology Configuration

1. **Node Addition:**
   - New nodes register with the network, receive an ID, and synchronize any topics that hash to their space.
   - Topics stored temporarily on other nodes migrate back to the appropriate new node.
2. **Node Removal:**

   - Nodes notify their neighbors of their departure, allowing data to migrate to remaining nodes.
   - Failed nodes are detected by the fault detector, which triggers replica reassignment.

---

5. Network Design

The **hypercube topology** is preserved but enhanced with:

1. **Dynamic Routing:** Routing tables adapt to reflect real-time node availability.
2. **Redundancy:** Each node maintains references to replicas stored within its neighborhood.

---

6. Data Flow and Operations

6.1 Topic Creation and Assignment

- Topics are hashed to a primary node and assigned to secondary nodes for replication.
- When the primary node fails, a secondary node is promoted to primary.

6.2 Message Routing

- Reads are directed to the nearest replica.
- Writes always go to the primary, with changes propagated asynchronously to replicas.

6.3 Fault Handling

- Failed nodes are detected through periodic heartbeat checks.
- Neighbors of failed nodes update routing tables and reassign topics.

6.4 Dynamic Changes

- Addition and removal of nodes trigger topology updates.
- Migrating topics maintain seamless access for clients.

---

7. Design Decisions and Tradeoffs

1. **Replication Strategy:**
   - **Decision:** Adopt primary-secondary for consistency.
   - **Tradeoff:** Consistency overhead may reduce write performance.
2. **Topology Adaptation:**
   - **Decision:** Dynamic updates based on runtime conditions.
   - **Tradeoff:** Increased complexity in routing and fault handling.
3. **Fault Tolerance:**
   - **Decision:** Prioritize availability over strict consistency.
   - **Tradeoff:** Temporary inconsistencies during synchronization.

---

8. Possible Improvements

1. **Advanced Replica Placement:**
   - Use machine learning to predict optimal replica locations based on usage patterns.
2. **Enhanced Fault Detection:**
   - Leverage distributed consensus protocols like Paxos or Raft for fault handling.
3. **Dynamic Load Balancing:**
   - Monitor real-time node usage to redistribute topics dynamically.
4. **Elastic Scaling:**
   - Integrate cloud resources to handle sudden load spikes.

---

9. Conclusion

The enhanced system builds on the foundational hypercube topology and DHT, offering **replicated topics** and **dynamic topology configuration** to meet modern requirements for scalability, performance, and fault tolerance. Future extensions can focus on predictive optimizations and integration with broader distributed systems.