Decentralised overlay network - Tapestry

Project Proposal - Team 26

Rohan Sridhar (2022101042) Mohammed Faisal (2022101101) Shreyansh (2022111002)

1. Project Title

Implementation of a Decentralized Overlay Network Inspired by Tapestry

2. Problem Statement

In large-scale distributed systems, efficient routing and resource location are critical challenges. Traditional fully connected networks are impractical due to high storage and maintenance costs, while unstructured peer-to-peer (P2P) networks suffer from inefficient search mechanisms. Tapestry, a structured overlay network, addresses these issues by providing scalable, fault-tolerant, and efficient routing with prefix-based matching.

This project aims to implement a decentralized overlay network inspired by Tapestry. The system will support efficient message routing, dynamic node membership, and resilience against node failures while ensuring fast lookups in a scalable network.

3. Framework and Technologies

- **Programming Language:** Go (Golang)
- **Communication Protocol:** gRPC
- **Data Structures:** Prefix-based routing tables
- **Hashing Mechanism:** SHA-1 or similar
- **Storage (Resources):** In-memory or simple file-based storage for node states

3.1. Reasoning Behind Technology Choices

- **Go (Golang) & gRPC**: gRPC provides high-performance, remote procedure calls (RPCs) with built-in support for error handling and serialization using Protocol Buffers. All of us have experience with gRPC in Go from the Assignment.
- **Prefix-Based Routing Tables**: The original Tapestry paper implements a prefix based routing system, using SHA-1, so we will be implementing the same.

4. Project Objectives

4.1. Implement Node Discovery & Routing:

- Nodes should be able to join and leave dynamically without breaking the system.
- Efficient routing using prefix-based forwarding. $O(\log n)$ hops for a routing request.

4.2. Resource Location & Lookup:

- Implement mechanisms for storing and locating resources.
- Ensure lookups occur in logarithmic time complexity.

4.3. Fault Tolerance & Adaptability:

- Handle node failures by reconfiguring routing tables.

5. Plan

5.1. Phase 1: Project Setup and Node Management

- Set up the development environment with Go and gRPC.
- Implement node initialization using SHA-1 hashing for unique IDs.
- Develop node join and leave protocols.
- Establish basic gRPC communication between nodes.

5.2. Phase 2: Routing Table and Message Routing

- Implement prefix-based routing tables as described in the Tapestry Paper.
- Implement the routing algorithm to ensure efficient routing in $O(\log n)$ hops.
- Test message forwarding between nodes.

5.3. Phase 3: Resource Lookup

- Implement the prefix based lookup algorithm from the Tapestry paper.
- Perform testing of resource storage and retrieval.

5.4. Phase 4: Fault Tolerance and Adaptability

- Implement heartbeat/etcd/similar checks using gRPC for node failure detection.
- Develop routing table recovery mechanisms for fault tolerance.
- Simulate node failures and test recovery.

5.5. Phase 5: Final Testing and Documentation

- Conduct comprehensive end-to-end testing.
- Optimize code for routing efficiency and fault recovery.
- Do performance scaling tests.

- Write the final report detailing design choices, implementation, and results.
- Prepare a short demo and presentation.

6. Deliverables

- Implementation of a Tapestry-inspired overlay network with routing, resource lookup, and fault tolerance.
- A written report detailing the implementation approach, technical challenges, and results.
- A presentation and demonstration showcasing the working system and its capabilities.