

Advanced Operating Systems CS 550

Programming Assignment 2

Design Document

Name: Harsh Shah	Hawk ID: A20582895
-------------------------	---------------------------

Table of Contents

Overview	4
Components	4
1. Indexing Server (indexing_server.py):.....	4
2. Peer Node (peer_node.py):.....	4
3. Configuration (config.json):	4
Design Trade-offs	4
Performance Analysis	4
Possible Improvements	4
Extensions	5

Overview

This document describes the design and architecture of a distributed pub/sub system using three main components: `indexing_server.py`, `peer_node.py`, and `config.json`. The system is designed to handle peer registration, topic management, message distribution, and scalability.

Components

1. Indexing Server (`indexing_server.py`):

- Manages peer registration and topic creation.
- Stores topics, peers, and messages.
- Handles client connections using `asyncio` for asynchronous I/O operations.
- Provides actions such as register, unregister, create/delete topics, subscribe to topics, send messages, and retrieve messages.

2. Peer Node (`peer_node.py`):

- Represents a client in the network that can publish or subscribe to topics.
- Manages its own server to handle incoming messages.
- Connects to the indexing server to register and perform operations like creating topics or sending messages.
- Uses `asyncio` for asynchronous communication.

3. Configuration (`config.json`):

- Contains IP addresses and ports for the indexing server and peer nodes.
- Allows easy modification of network settings without changing the code.

Design Trade-offs

- **Asynchronous I/O:** Chosen for scalability and performance under concurrent connections. This allows handling multiple peers efficiently but adds complexity to error handling and debugging.
- **JSON-based Communication:** Simplifies message parsing but may introduce overhead compared to binary protocols.
- **Dynamic Port Allocation:** Peer nodes dynamically find available ports, which simplifies deployment but may lead to conflicts if ports are scarce.

Performance Analysis

- **Scalability:** The system shows good scalability up to 8 peers with linear response time growth. Beyond this point, performance may degrade due to increased coordination overhead.
- **Throughput:** Achieves high throughput with 8 peers, peaking at around 5000 operations per second.
- **Latency:** Latency increases with more peers but remains manageable within expected operational limits.

Possible Improvements

1. **Enhanced Error Handling:**

- Implement more robust error handling mechanisms for network failures and message parsing errors.
- 2. **Load Balancing:**
 - Introduce load balancing across multiple indexing servers to distribute the load more evenly.
- 3. **Security Enhancements:**
 - Add authentication and encryption to secure communication between peers and the server.
- 4. **Persistent Storage:**
 - Use a database for storing peer information and messages persistently instead of in-memory storage.
- 5. **Monitoring and Logging:**
 - Implement comprehensive monitoring tools for real-time performance tracking and logging enhancements for better diagnostics.

Extensions

- **Web Interface:** Develop a web-based interface for easier management of topics and peer interactions.
- **Advanced Message Filtering:** Implement filtering capabilities based on message content or metadata for more efficient subscription management.

This design provides a flexible foundation for a scalable pub/sub system with room for future enhancements in performance, security, and usability.