P2Pal - Programming Assignment - Part 1

Introduction

In this lab, you will start building a small peer-to-peer application called P2Pal.

Core Requirements

Basic GUI Implementation

Your implementation must create a chat window using Qt6 with:

- A chat log area displaying messages

- A text input area for entering new messages

- Support for multiple lines with word wrap

- Auto-focus on the text input area when launched

Network Communication

Implement UDP-based messaging using QUdpSocket:

- Message serialization/deserialization using QVariantMap
- Support for local port discovery
- Basic message format containing:
    - ChatText: The actual message content
    - Origin: Unique identifier for each P2Pal instance
    - Sequence number: For message ordering

Gossip Protocol

Implement two key mechanisms:

1. Rumor Mongering:
    - Messages must be uniquely identified using two components:
        - Origin: A unique identifier for each P2Pal instance
        - Sequence number: Starting from 1, incrementing each message from that origin
    - Messages should be propagated in sequence number order (e.g., send message 3 before message 4)
    - Implementation requires:
        - Setting timers (1-2 seconds) using QTimer

        - Resending messages if no response is received
        - Tracking which peers have which messages
2. Anti-Entropy:
    - Peers periodically compare their message histories
    - Use vector clock concept where each node tracks:
        - Which messages it has seen from each origin
        - Up to what sequence number for each origin
    - Example: If peer A has seen messages from C up to sequence 5, and peer B only has up to sequence 3, then A knows to send C's messages 4 and 5 to B
    - Ensures reliable message propagation across multiple hops

Peer Discovery

- Implement neighbor discovery on local ports
- Support dynamic peer addition via IP/hostname
- Maintain connections with immediate neighboring ports

Technical Requirements

- Use a systems language (C, C++, Rust, etc.)
- Qt6 framework
- Use Build system, provide necessary documentation
- Git for version control
- Support for running multiple instances locally
    - Script to launch multiple instances etc

Testing Requirements

Verify your implementation by:

1. Running 2-4 instances locally
2. Testing message propagation
3. Verifying message ordering
4. Testing peer discovery

Submission Process

Submit your work through Canvas:

- Submit a zip file from your git repository

- Include comprehensive documentation

- Provide build instructions

- Include basic test cases

- Provide scripts to automate deployment and testing