CS 550 Programming Assignment 2

General Instructions

- Due by 11:59pm, Tuesday, March 11, 2025
- Late penalty: 10% penalty for each day late.
- This is an individual assignment. Although you may work in groups to brainstorm on possible solutions, your code implementation, report, and evaluation must be your own work. Upload your assignment on Canvas with the following name: Section_LastName_FirstName_PA1.zip
- Please do NOT email your assignment to the instructor or TA!
- Use any popular language, if in doubt ask the TA ASAP.

Introduction

Enhance your P2Pal implementation from Lab 1 to:
1. Implement Destination-Sequenced Distance-Vector Routing (DSDV) routing for point-to-point messages
2. Enable NAT traversal via route rumor propagation

Part 1: Routing Implementation

Implementation Requirements:
1. DSDV Routing Table 2. Route Rumors 3. Private Messaging

1.1 DSDV Routing Table

Update Rules:
1. When receiving any Rumor message:
`if (message.SeqNo > currentSeqNo[message.Origin]) {         routingTable[message.Origin] = (senderIP, senderPort);     }` 2. Forward all Rumor messages to random neighbor
3. Maintain sequence numbers per origin

Example Flow:
1. Node A (1.2.3.4:43433) sends Sequence #23 → Node B
2. Node B updates: `3. Node B forwards to Node C   4. Node C updates:`

1.2 Route Rumors

Implementation Checklist:
- [ ] Add a timer for automatic route announcements:
example: `QTimer::singleShot(60000, this, &P2Pal::sendRouteRumor);` - [ ] Generate route rumors with format:

`{ "Origin": "NodeID", "SeqNo": N }` - [ ] Send initial route rumor on startup
- [ ] Process route rumors same as chat rumors (no GUI display)

1.3 Private Messaging

GUI Requirements:
- Visible node list with recent routes
- Private message dialog on node selection

Message Format:
Example:

```
QVariantMap msg;
msg["Dest"] = "NodeB";    // QString
msg["Origin"] = "NodeA";  // QString
msg["ChatText"] = "Hello";// QString
msg["HopLimit"] = 10;     // quint32
```

Forwarding Logic:
Example:

```
if (msg["Dest"] == localID) {
    displayMessage(msg);
} else if (msg["HopLimit"] > 0) {
    msg["HopLimit"] = msg["HopLimit"].toUInt() - 1;
    sendTo(routingTable[msg["Dest"]], msg);
}
```

---

2. Part 2: NAT Traversal

2.1 Testing Environment Setup

Linux Network Namespace Approach:
You need to create multiple NAT environments to simulate this part of the assigment. To do so follow the instructions below

```
# Create NAT environments
sudo ip netns add nat1
sudo ip netns add nat2

# Configure different NAT types
sudo ip netns exec nat1 iptables -t nat -A POSTROUTING -j MASQUERADE
sudo ip netns exec nat2 iptables -t nat -A POSTROUTING -j MASQUERADE --random
```

2.2 Rendezvous Server

-noforward Mode Implementation:

```
if (noforwardMode && message.contains("ChatText")) {
    return; // Do not forward chat messages
}
else {
    forwardMessage(message); // Forward route rumors
}
```

2.3 NAT Hole-Punching

Key Modifications:
1. Add to all rumors:
`msg["LastIP"] = senderIPv4Address;    msg["LastPort"] = senderPort;`
2. Automatically add discovered public endpoints
3. Route preference rules:
`bool isBetterRoute(const Route &old, const Route &new) {        return (new.seqNo > old.seqNo) ||                (new.seqNo == old.seqNo && new.isDirect);    }`

Testing & Validation

Core Functionality Tests

```
# Run 3 nodes locally
./p2pal -port 12345 &  # NodeA
./p2pal -port 23456 &  # NodeB
./p2pal -port 34567 &  # NodeC
```

Verify:
- Route rumors propagate within 60s
- Private messages route through intermediates

NAT Traversal Tests

```
# Rendezvous server
./p2pal -port 45678 -noforward &  # NodeS

# NATted nodes
ip netns exec nat1 ./p2pal -port 11111 -connect 45678 &  # NodeN1
ip netns exec nat2 ./p2pal -port 22222 -connect 45678 &  # NodeN2
```

Verify:
- NodeN1/N2 discover each other's public endpoints
- Direct messaging works despite NodeS's -noforward

Submission Requirements

Documentation Checklist

- ☐ Clear build/run instructions
- ☐ Network namespace setup guide
- ☐ Testing procedure explanation
- ☐ Known limitations section
- ☐ Create Zip file and submit!