Indian Institute of Technology
Dharwad

Computer Neworks

CS-348: Project Report

Peer-to-Peer VoIP Platform

Submitted To:
Prof. Tamal Das
Department of Computer
Science and Engineering

Submitted By:
Aman Pushkar - MC22BT004
Tanishque Soni - MC22BT025
Vikas Jain - MC22BT029
Adarsh Gupta -MC22BT001
Vivek Kumar -CE22BT015

Contents

Abstract

This report details the development of a lightweight peer-to-peer platform enabling voice over IP (VoIP) calls between users. The platform utilizes Python's socket programming capabilities along with threading for concurrent operations. This document outlines the project's motivation, goals, technical architecture, implemented features, and instructions for running the application. A demonstration of the platform's functionality is available in the accompanying screen recording. The system emphasizes decentralized design and user simplicity, making it a suitable solution for small-scale communication.

1 Introduction

In today's interconnected world, efficient and direct communication is paramount. This project aims to create a simple yet functional peer-to-peer (P2P) platform that allows users to make voice calls directly with each other, without relying on centralized servers for media. This approach offers potential benefits in terms of resilience, and reduced infrastructure costs for small-scale communication needs. The system also serves as a practical demonstration of networking principles and real-time communication.

2 Motivation

The primary motivation behind this project is to explore the principles of peer-to-peer networking and implement a practical application demonstrating its capabilities. Specifically, we aimed to:

- Gain hands-on experience with socket programming in Python
- Understand the challenges and solutions involved in establishing direct communication between network nodes
- Develop a functional VoIP system for real-time voice communication
- Create a lightweight and easily deployable communication platform
- Study concurrency and synchronization in a real-world networking context

3 Goals

The main goals of this project were to:

- Implement peer registration and discovery using a simple tracker server
- Enable users to initiate and receive VoIP calls with other registered peers

- Ensure the platform is relatively easy to set up and run on different machines
- Provide a clear demonstration of the implemented features
- Maintain low latency and minimal resource usage for real-time communication
- Support concurrent operations using multithreading for seamless user experience
- Design a modular codebase to allow easy extension or integration of additional features
- Handle basic error scenarios and ensure connection stability during communication
- Emphasize simplicity, portability, and minimal configuration in deployment

4 Technical Stack and Requirements

4.1 Core Technologies

- **Programming Language:** Python 3.x
  - Required version: Python 3.7 or higher
  - Reason: Modern socket programming capabilities and threading support
- **Essential Libraries:**
  - `pyaudio`
    - Purpose: Real-time audio streaming
  - `socket`
    - Purpose: Network communication
    - Protocols: TCP for registration, UDP for voice/file transfer
    - Port Configuration:
      - Base Port: 7000 + PID offset
      - Signaling Offset: +2

4.2 System Requirements

- **Hardware Requirements:**
  - Microphone (built-in or external)
  - Speakers/Headphones

  - Network Interface Card

- **Network Requirements:**
  - Stable Internet connection
  - Open UDP/TCP ports

4.3 Installation and Setup

```
# Install required packages
pip install pyaudio
```

4.4 Configuration Settings

```
# Network Configuration for peer.py
TRACKER_IP = 'Tracker IP_Address'
TRACKER_PORT = 6000
PEER_PORT = 7000 + os.getpid() % 1000

# Audio Settings
CHUNK = 1024
FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 16000
```

4.5 Running the Application

1. Ensure all prerequisites are installed
2. Configure network settings in the code
3. Run the tracker server first
4. Execute the peer application:

```
    [Linux] python3 peer.py
    [Windows] python peer.py

```

5 Code Functionality

5.1 tracker.py Implementation

The tracker.py implementation establishes TCP connections with the peer for peer registration and updates. It sends the list of ip addresses and the port number of all

the peers in the network. It sends the updates peer list whenever a peer exits from the network.

5.2 peer.py Implementation

peer.py follows a similar architecture but uses colon-delimited messages for signaling and implements thread-safe call management through a Queue object. While maintaining the same core VoIP functionality, it features more robust error handling in voice call operations and uses different port offset values. Despite these internal implementation differences, it exposes the same command-line interface for peer interaction and call management ensuring consistent user experience across all peers.

6 Demo Video

- Project Demo Video: https://drive.google.com/drive/folders/1D8ZvfESpLHoM_P_NdoVtQoIJqXxEIwaL?usp=sharing

7 Features Implemented

The platform includes the following key features:

- **Peer Registration**: Peers register with a central tracker server
- **Peer Discovery**: Retrieve list of registered peers
- **VoIP Calls**: Initiate and receive voice calls
- **Real-time Communication**: Audio streaming via UDP

8 Implementation Details

8.1 Running the Tracker (tracker.py)

1. Save `tracker.py`
2. Run: `python3 tracker.py`
3. Tracker will start listening for connections
4. Tracker must run on Linux.

8.2 Running the Peer Application

1. Save `peer.py`

2. Run: `python peer.py`
3. Each instance will register with the tracker

8.3 Usage Instructions

Option	Function
1	List available peers
2	Start a call
3	Exit application

Table 1: Peer Application Menu Options

9 Conclusion and Future Work

This project successfully demonstrates fundamental peer-to-peer (P2P) networking concepts through the implementation of real-time VoIP calling. It showcases practical use of Python's socket programming and multithreading to build a lightweight, decentralized communication platform.

During development, we also explored implementing a group chat feature. While the initial framework for group messaging was partially developed, we faced integration challenges and encountered some errors. Due to time constraints, we decided to prioritize and finalize the VoIP functionality for this version.

Future enhancements could include:

- Enhanced error handling and connection recovery
- Text messaging capabilities with group chat integration
- More intuitive and user-friendly interface
- Advanced peer discovery mechanisms
- Security features such as end-to-end encryption
- Support for group voice calls
- Extension to group video calling functionality
- Cross-platform compatibility and mobile support

With these improvements, the platform can evolve into a more comprehensive and scalable communication tool suitable for a broader range of applications.

References

[1] PyAudio Documentation.
PyAudio Library.
https://people.csail.mit.edu/hubert/pyaudio/

[2] Python Socket Programming.
Python Documentation.
https://docs.python.org/3/library/socket.html

[3] Peer-to-Peer Networking Concepts.
Wikipedia.
https://en.wikipedia.org/wiki/Peer-to-peer