The Implementation of an Innopolis Video Call Application Using Python Sockets

Rafik Hachana, Parth Kalkar, Igor Mpore
GitHub Repository

May 2021

Abstract

We describe various implementation aspects of a video network call by using Python sockets. The call is implemented as a peer-to-peer application and relies on TCP communication (except for a shared database that uses a client-server model). The application's video call only works for non-NATed peers, i.e. the call correspondents should either have a public IP address, or be in the same LAN. For the time being, the application only works in Linux environments.

1 Introduction

The project consists of the implementation of a peer-to-peer (P2P) video calling application (that supports both video and audio calls) using TCP sockets in Python. In order for a peer to call another peer, they should both be in the same private network or both should have a unique public IP address. As opposed to other P2P applications, this application does not distinguish a master peer that synchronizes other nodes, instead we use a remote shared database that stores usernames and their corresponding IP addresses. For now, the application does not use any security measures like encryption or authentication, that is due to the time constraints and the primary goal being a simple MVP that shows the basic functionality.

The application supports a quick registration each time a user opens it, with a username and no password. Once the user is registered he can call online users or receive a call. For the time being, the application can only run in Linux environments. However, it can be easily adapted and packaged for devices running Windows or MacOS.

In this report, we will get an overview of the implementation of the application, including the technologies used and the protocols designed for P2P communication, then we will offer a short installation and user guide, and finally we will discuss potential features that can be added and the challenges related to them.

2 Implementation details

In this section, we will go over the implementation details of the application, we will discuss the technologies used and how we used them to make different components, and we will explain how we implemented the P2P communication.

2.1 Technologies used

The technologies used can be classified by the component they belong to. The app has 2 main components: The frontend UI and the core of the application, as well as 2 external components consisting of 2 databases: The Redis local in-memory database and the MongoDB remote database.

2.1.1 Technologies for the application core

Python : Python was used to program the core of the application. Python was chosen primarily because of its simplicity which reduces the required development time, and also because of the team's profile. Many native python libraries were used, such as the `socket` library for implementing the TCP sockets, the `pickle` and `json` libraries for converting objects to bytes or strings before sending them over sockets, the `time` and `datetime` libraries for getting time and sleeping, as well as the `threading` and `multiprocessing` libraries for making processes and threads.

OpenCV library : The OpenCV library was used for capturing frames from the user's camera, and resizing and encoding them.

PyAudio library : The library was used for capturing and playing audio.

PyMongo library : The library was used for connecting to the remote MongoDB database. The library also requires the installation of the `dnspython` library that is used to connect to the database.

Redis library : The library was used to connect to the local Redis library, used for inter-process communication.

2.1.2 Technologies for the frontend UI

HTML, CSS and JavaScript : The UI was implemented using vanilla HTML, CSS and JavaScript, the same way that websites are implemented.

Electron.js package for Node.js : The Electron JavaScript framework was used to make the implemented web interface into a desktop application. It requires the use of a Node.js server.

The Redis package for Node.js : The Redis package was used to write to the Redis in-memory database.

2.1.3 Technologies used for databases

MongoDB : The application uses the MongoDB NoSQL database for storing the user's username, IP address and online status once the user is registered. The database is hosted on a remote distributed server cluster offered by MongoDB's Atlas service.
Redis : Redis is a very fast in-memory database, it is used for inter-process communication between the different components of the application. We can consider it as a shared memory space where all components can read or write data.

2.2 An overall application organization

The application's UI is considered as one component, but the core has many individual components. All components communicate through Redis. A diagram of their relationship is shown below:
We also present the flow of the UI and the core as FSAs in the following diagrams:

2.3 Peer implementation

Here we mention the processes running on each peer (the call listener, the manager process watching the database, and the call maker). Once that is described, we talk about the call components that are started in the call.

2.4 Peer-to-peer communication

We mention the protocols we designed.

3 Installation and usage guide

The application can be downloaded and installed on any Linux machine. The steps are as follows:

3.1 Installation guide

1. Download the .zip file of the project here.
2. Extract the .zip file to the location where you want the app to be installed.
3. Open the terminal in the application's directory and run the command `bash install.sh`. This will install all the dependencies and requirements for the application.

3.2 User guide

1. Open the terminal in the application's directory and run the command `bash run.sh`.
2. Once the application's window opens, enter your username and click on "Save". Now you will be able to see the list of online users.
3. If you want to call a user, press the call icon next to their name.
4. Once the other user accepts the call, you will be able to see and hear them.
5. During the call you can toggle your microphone and webcam using the corresponding buttons.
6. To finish the call, use the corresponding button. You will be brought back to the home screen.

You can find the screenshot for the guide here.

4 Future extensions

1. Security
2. Text chat
3. Password authentication
4. Error handling and messages
5. NAT traversal
6. Fallback relay server
7. Video and audio performance

5 Conclusion