Sync

...A Music Synchronizer

Course Project for CS311

Trimester 1 (2020-21)

Submitted to :

Dr. Ravi Bhandari

Submitted by:

Devin Garg (B18CSE011)

Kartik Vyas (B18CSE020)

Contents

1 Introduction

Music forms an important part of the daily lives of a large group of the population. When there is any cause for celebration whether it is an after-hours work place scene or a students' hostel, people can be in a pickle if a good music system is not at hand. To improve upon such a situation, in this project we have implemented the concept of sockets to create an application that can be used for music synchronization across devices connected to the same local network! We have used python sockets as a means to establish a connection between devices that are connected to the network.

1.1 Motivation

As mentioned above, the main motivation behind this project was the need for a make-shift common music system. When coupled with the content of this course, it seemed like a perfect overlap between a use-case and the concepts we were going to learn about in communication and networking.

2 Problem Description

Briefly it can be described as playing a track over different devices in synchronization to obtain surround sound effects. Considering the technical aspects, there would be a network of devices and the clients would send track suggestions to the server and then the server may or may not have the track. In case, any of the clients has it, the track is transferred to all the endpoints. The user then may play, pause, stop or end the track player.

3 Related work in the area

This problem has wide implications and usage in real life. Its importance is well understood in the market and there are some mobile apps as well which now allow the functionality of synchronizing tracks. Though, this area has only been explored in limited depth. Adding functionalities like suggesting songs, song transfer makes the project interesting and exciting. The apps that are currently available in the app stores are not open source and contributing for open source is motivating and with time, it will result in something that will be of usage for masses.

The course content forms the perfect base for understanding the fundamentals of the project and the course helps in holding grip and better understanding of concepts used in the project such as connection of devices, their interaction by communication and file transfer and then performing tasks in synchronization. By using sockets fundamentally, the users do not need internet connection if the track is already available in any of the devices. The work would be highly scalable.

4 Solution

Before diving into the details of our implementation in this project, let's briefly take a look at what sockets are and how they function.

4.1 Sockets - the answer!

Simply put, a socket is an endpoint communication instance for a node that is present in a network. In general, sockets include information about the transmission protocol in use, the IP address and the port number. So, how that works is - whenever a node acts as a server, it opens a socket and starts listening for connection requests. A client on the other hand is aware

of the IP address of the server and the port on which the socket is open. With this information, the client is able to try to connect with the server, which, given some constraints are satisfied which may include authentication, then accepts the connection request and then a connection is established between the client and the server.

4.2 Our Project

In this project we have established a socket connection between several laptop/PCs to facilitate synchronization of a particular sound track. Several clients can get connected to the server and then they can suggest the track that they wish should be played. This suggestion would be sent to the server. Afterwards, once all the clients are connected and their track suggestions are received; the server decides the track that is to be played. In case, the server endpoint does not have the track, it checks if any client has the same, in case they do the track is sent to the server (and if none of the endpoints has the track, it obviously cannot be played). Once the server has the track file, it checks if all the client endpoints have the track or not, in case the track is not at any of the endpoints, the server transfers the file to the respective client. Then the user may play, pause and stop the player accordingly.

The project is implemented using the socket library of python. Pygame library accounts for the track player and PyQt5 has been used extensively for building the GUI of the program.

4.2.1 Codeflow / Pseudo Code

1. Server creates a socket connection
2. Server enters the number of client servers to be added
3. Clients enter the name of the server endpoint and join
4. Clients suggest tracks that they want to be played
5. Server receives the suggested tracks and does tracks selection
6. If track is not present in the server endpoint :
    a. Check if the track is present in any client endpoint by running a loop
        i. If yes, transfer the song to the server endpoint
7. If the track still could not be obtained by the server :
    a. Return that the track is not available on any of the clients
8. Check if the track is not present in any client endpoint by running a loop :
    a. If not present, transfer it to the concerned client endpoint from the server endpoint.
9. Now every endpoint has the track and using pygame and by socket communication the server endpoint can play, pause, stop, or end the track player.

4.2.2 Snapshots of the demo

Following are the screenshots from the project. We have implemented the project using the command line interface. Also, a GUI has been built (with almost common functionality).

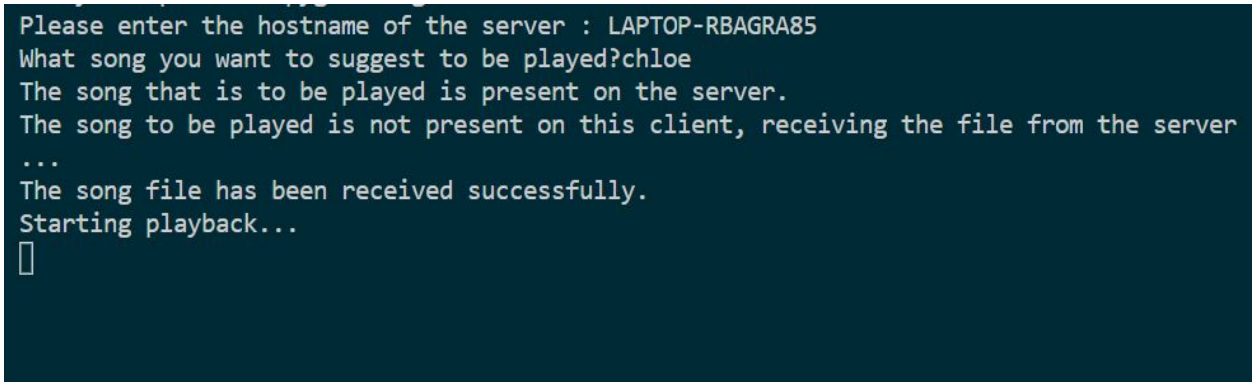
4.2.2.1 Command Line Interface

The command line interface has two components - client and server. The working of both the components are shown below.

Server Endpoint :

```
ew folder (2)\DC\file\server> pyt
hon server.py
pygame 1.9.6
Hello from the pygame community.
https://www.pygame.org/contribute
.html
LAPTOP-RBAGRA85
How many clients do you want to a
llow?3
Connected with ('192.168.56.1', 5
5041), allotted ID=0
Connected with ('192.168.56.1', 5
5042), allotted ID=1
Connected with ('192.168.56.1', 5
5043), allotted ID=2
Following are the song recommendations by the client servers
stay
chloe
stay
Enter the song to be played (followed by the file extension)stay.mp3
Checking if server end point has the song.
Song is already present on the server endpoint
Veryifying whether all the clients have the song.
Transferring song to  ('192.168.56.1', 55041)
Song transferred to  ('192.168.56.1', 55041)
Transferring song to  ('192.168.56.1', 55042)
Song transferred to  ('192.168.56.1', 55042)
Transferring song to  ('192.168.56.1', 55043)
Song transferred to  ('192.168.56.1', 55043)
What do you want to do? (play/pause/resume/stop/end)play
Starting playback...
What do you want to do? (play/pause/resume/stop/end)
```

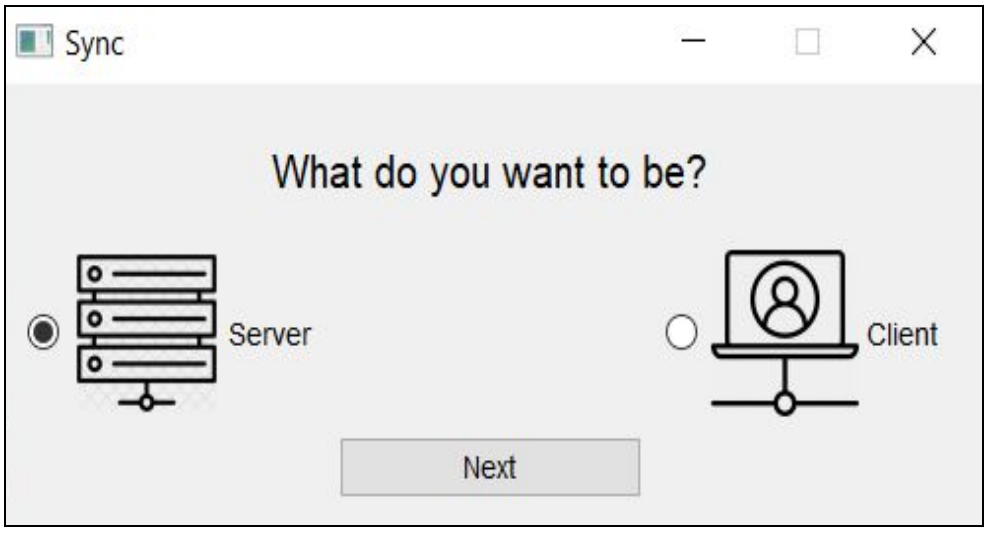
Client Endpoint :



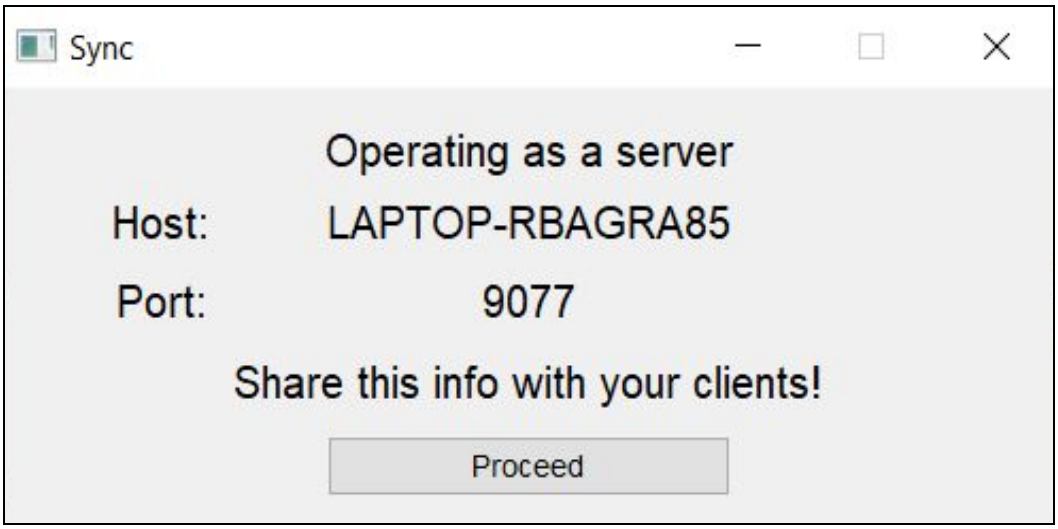
The first snip is of the server endpoint and it first shows the hostname and then enters as the number of clients to be added. It then decides the track to be played after receiving the suggestions and since the three clients do not have the track, it is sent to all the three clients and then played. The second snip is of the client endpoint, it enters the name of the host to which connection is to be made. Then, it suggests a track and then it receives the track from the server as it was not earlier present. Then, the playback is started after receiving command from the server.

4.2.2.2 Graphical User Interface

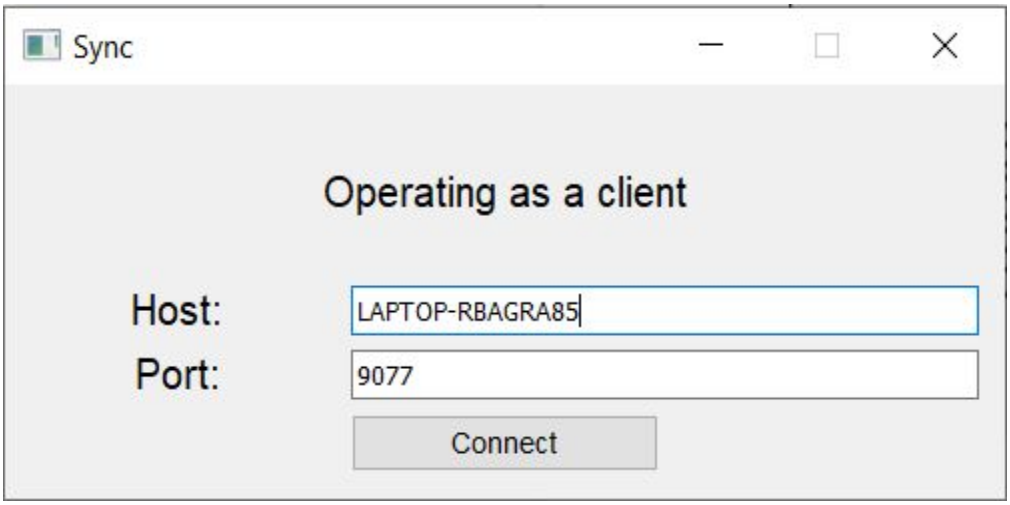
The following page(s) show the GUI that is built using PyQt library of python. The functionalities of each snip is mentioned right below them.



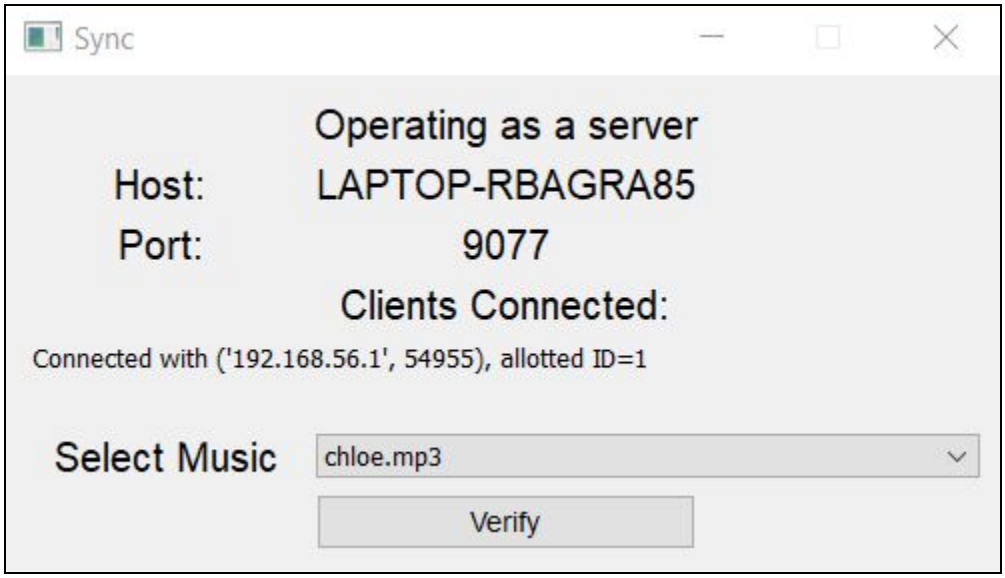
The first screen asks what the node wants to be - the server or a client



The first interface for the server endpoint, showing the details required by the clients for connecting and the proceed button to go to the next window where we pick the sound track to be played.

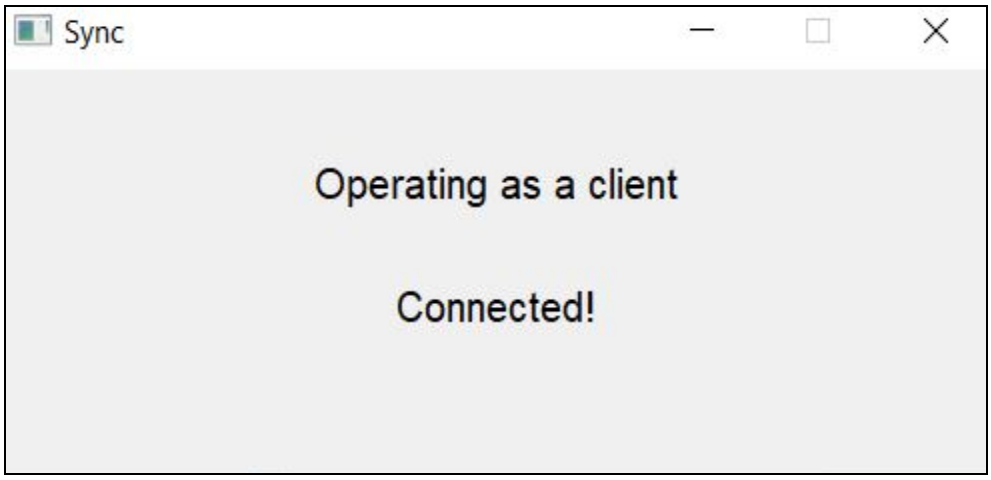


The first interface of the client endpoint where the client inputs the server's name and port where it is listening to establish a connection.



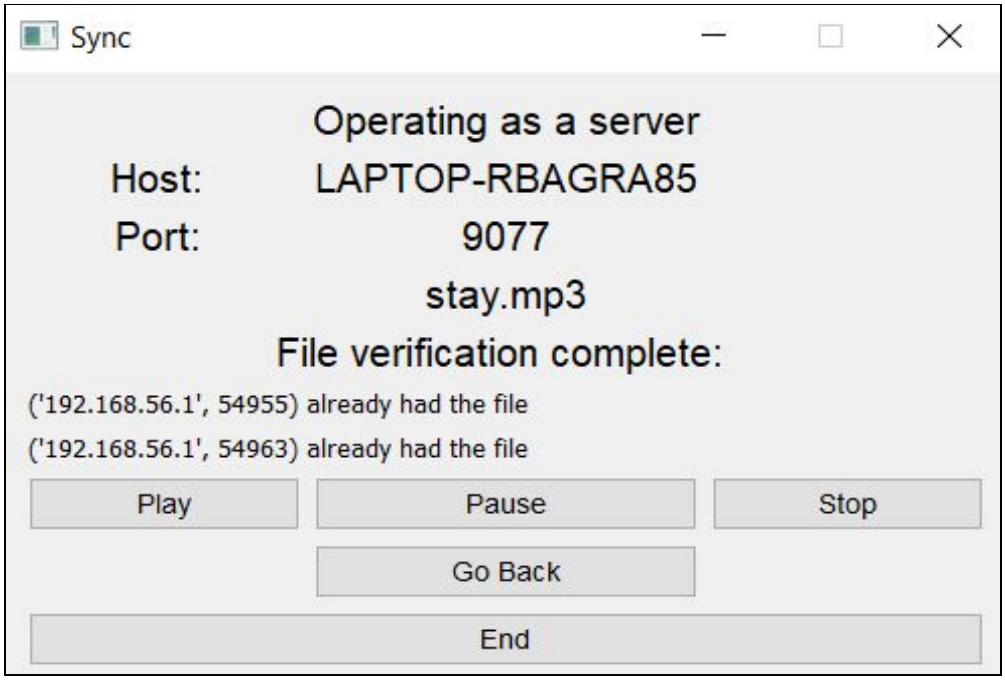
The user at the server node selects the track from the drop down menu and then verifies if the track is present with every client or not. In case not, the track is transferred to the clients that need it.

(Note: The songs listed in this drop down list are the ones that are present with the server.)



The client interface which is shown after successful connection.

From here on out, the client window shows this message while the music played on it is controlled by the server.



The final window at the server side which is the track player, it offers utilities of play, pause, stop or end the player. One may also go back and select some other song to play.

4.2.3 Hiccups in the process

As we began working, the major thing to look into was the working of sockets and music playback. We turned to the inbuilt socket library that in turn contacts the socket API of the OS. The understanding of how socket commands work was a major point. Most of those being blocking - in the sense that they freeze the python process until they go through with the respective operation. Music playback was sorted with the help of the *pygame* library.

The next major thing to look at was synchronization across devices. This meant taking into consideration the lag that was introduced while sending the command to play/pause/stop the music and the time it took to take the respective action. To estimate what needed to be the delay to be introduced at the server node, we employed the *ping* tool.

```
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=37ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=2ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=3ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=70ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=79ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=91ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=101ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=115ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=119ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=133ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=139ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=151ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=162ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=168ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=180ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=192ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=206ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=210ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=18ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=27ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=44ms
Reply from fe80::f43f:e7be:d8a8:c57e%18: time=55ms

Ping statistics for fe80::f43f:e7be:d8a8:c57e%18:
    Packets: Sent = 46, Received = 46, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 213ms, Average = 99ms
```

As shown in the above image, we found out the average delay in the case of using a python socket which came out to be around 100ms. We introduced this delay in the server node to keep the nodes in sync. Note that this was under test for a specific case (though tested multiple times).

4.2.4 Codebase Link

https://github.com/vyaskartik20/SYNC

5 Contribution

Work	Devin	Kartik
Socket Communication	Major	Minor
File Transfer	Minor	Major
Playback	Minor	Major
GUI	Major	Minor

6 Future Prospects

The wide real life application makes the project highly scalable which can be extended to meaningful end products. Possible future work that can be done :

- ❖ Mobile Interface
- ❖ Peer to peer file transfer
- ❖ Playing track that is available over web

- Track polling utility by the clients
- Authentication
- Including automatic pinging and delay setting

7 References

- Front page image - CC0 1.0 (Source)
- https://docs.python.org/3/library/socket.html
- https://realpython.com/python-sockets/
- https://docs.python.org/3/howto/sockets.html
- https://www.pygame.org/docs/ref/music.html
- Images in the GUI -
  - https://thenounproject.com/term/client/
  - https://vectorified.com/mainframe-icon