Music Hub

Junghun Kim
University of California, Irvine
junghuk@uci.edu

Minhaeng Lee
University of California, Irvine
minhaenl@uci.edu

Abstract

In this report, we introduce a novel way of music content sharing method. While previous approaches to share music focus on file content or distribution, our method is focusing on play. The most important concept of our sharing method is playing simultaneously. We embedded time stamp on each music packet to meet pace of different devices and during playing time, different speakers equipped on different devices work as a single speaker. Moreover, we construct region based music play that means each device is controlled based on its location from access point (AP). We demonstrated our device with two raspberry pis and their connected speakers and on mac book.

1. Introduction

1.1. Motivation

A number of romantic movies have their own special scene to touch people’s heart. Almost always, those scene background music. The music is one of the important part to those scene romantic. However, do those romantic scenes happen only in the movie? The moment much more romantic than movie is our real life. We imagine that to put some background music in our real life.

However, broadly, there are two practical obstacles. Firstly, if we make our personal background music by installing public speakers. Installing those speakers is one of big problem to solve though, if we actually install public speakers and use for background music, then it may become public noise problem. Obviously, there will be some people who don’t like to listen whatever we play. The point is that the music should be played only for people who need to listen. The second problem is natural way of playing. If we notice the background begins to play at certain moment, then the touching might be shrunk. In other word, the music should be played without our certain consciousness. In summary, the state overcome practical obstacles is that the atmosphere where a music played only for intended person without consciousness just like background music in movies.

1.2. Technical Background to solve the problems

Our imagination is ready to be realized with technical development. As technology have been developed, different types of wearable devices are released. Those devices have their own operating system (OS) and large amount computation power. Recently released one is smart watch. One of the popular one is iWatch. It has heartbeat meter, thermometer and have a number of feature compatible to smart phone. After the moment when smart phones are distributed to us, it is another forward step to new technology. Next expected wearable device is smart headphone.

Many people say smart phone around us already cover almost every functionality of wearable devices in future. However, each device has its own features. For watch type of device has heartbeat and thermometer as its main feature. Those are only useful for devices which is attaching to user’s skin. For smart phone, it has its own features. The main property is that a speaker is attached on user’s ears when a user wears it. Even though smart phone has its own speaker though, anytime attaching (smart headphone) or selective attaching (smart phone) is fundamentally different.

In this paper, we developed a prototype of those smart headphone which consists of body, speaker, and battery. The body is Reaspberry pi B+ and it has Rasbian based on...
Figure 2. Connection types. In simple broadcasting system (a), the host device only works as sender and clients work as receiver. However, in our system (b), each device works as both server and receiver simultaneously.

Figure 3. Developed system design.

Linux as its OS. For demonstration, we use total three pair of devices: two of them is raspberry pi and the other is laptop computer.

2. System Design

We display our simplified system design in Figure 1. As we described before, each machine designed based on raspberry pi. The server and clients are connected via WiFi. Simply say, the music is streaming via WiFi into clients and then the clients play the music with exact sync with others music. As we illustrated in Figure 2, even though, the functional role of a server and a client is different, each device can act as both a server and a client even at the same time.

More developed version is shown in Figure 3. Each smart headphone is designed based on raspberry pi and connected through WiFi. The difference to simple version is that it works based on its region. If a device is included in certain region then it turns on, otherwise turns off. Turn on/off doesn’t mean actual power but mean specific action based on our design.

3. Connection Types

There are roughly two different connection types we include. Working like radio, it just streaming for every devices closer enough to AP. Another possible type is that working when certain condition is satisfied such as a number of client.

(a) Direct/Individual Streaming
(b) Lazy/Mass Streaming

Figure 4. Two types of streaming.

Direct/Individual Streaming: The most simple type of streaming (Shown in Figure 4 (a)). Once a client sends connection request to server, then the server (host) answers as streaming suggestion. The hand shake is achieved, then the client ask streaming request and then actual streaming from host to client begins. One big feature of this method is simple but cannot consider mutual condition based on client. The server can see only one client at single streaming.

Lazy/Mass Streaming: In contrast to direct streaming, lazy streaming involves more than one client for one streaming. As shown in Figure 4 (b), for each client, the 4-way hand shaking frame is similar though, the host holds streaming packet until certain condition is satisfied. In the figure, it has minimum number of client constraint. The host wait until the number of streaming responses meet certain number (2 in here) and then start streaming. This way of streaming begins play at certain condition and stops at another condition.

4. Implementation

In this chapter, we explain actual implementation how to make our prototype and code our software. As shown in Figure 5, it has three different parts: body, speaker, and battery. Especially, the body - raspberry pi - has WiFi dongle to connect AP. It is mobile device, so battery installation is inevitable.

We developed software from scratch using Java. Java is optimized in terms of various usage for embedded device so we selected it for the implementation. Where ever Java virtual Machine (JVM) is installed then our software can
Figure 5. Our prototype. Basically, the prototype is comprised of three devices: Raspberry Pi, speaker, and battery. Raspberry Pi has WiFi dongle to connect AP.

Figure 6. Illustration how to sync music. Time stamp when a given packet have to be played is embedded in each packet. Each time, our system check the time stamp, and choose one of possible action: wait, play, drop. It work properly.

4.1. Music Sync

The most important feature is music sync. To guarantee music play at the same time, we embedded time stamp to each music packet. The size of time packet can change based on broadcaster though, it should be larger than time stamp to work effectively. If it is too small, the efficiency will be reduced.

We display an illustration of time stamp embedding and how it handle in client side in Figure 6. Each packet is sent to a client with time stamp. The client check current time which is already synced to server and if time stamp is before current time then wait, match perfectly then play, lastly, already passed then drop the packet. It is quite straightforward and The second case rarely happens and usually wait and play. The packet size highly related to efficiency though, small packet size can have high reaction velocity.

4.2. Modules

In our software implementation, there are two layers and in the second layer there are five modules. The outermost layer is our device Raspberry Pi with its operation system Raspbian. We install Apache Tomcat which works on JVM to control the device via web. Actually, in Tomcat layer, we have five modules.

**Timelookup:** At the beginning, we sync the local time of each client to server. If Internet is connected, it checked global time from official web site and otherwise, clients just follow their server time.

**Broadcaster:** A different name of server. It controls music file read from both local or another server, dividing packets, embedding time stamp and broadcasting.

**Player (Client):** A module that receive packets from server and play music through its speaker line.

**WiFi-Detector:** Our prototype works on region based constraint, we need to check the position of each client in real time. Technically say, it cannot measure the distance to other clients though, we approximate the distance to other clients or AP from the strength of WiFi signal.

**Web-Based GUI:** Since we don’t have display in our prototype, the only way to control the device is remote controlling. Instead of command line typing, we make fancy web-based controller.

5. Embedded issues have to solve

Since our prototype is broadly classified as embedded system, we cannot ignore involved important issues. Usually mentioned issues are capacity, energy efficiency, and security.

**Capacity:** The most of the computation unit of embedded system is weaker than our laptop, desktop and workstation. Moreover, each system has different computation power, some of them are weaker than the others. But entire system serving for them should consider to serve for every devices. In our case, majority of devices have enough computation power though, for minority that has less power than others,
we consider dynamic adaptive streaming. That means streaming quality depends on client’s status. For example, a server service low-quality streaming for weaker machine and high-quality for powerful one.

**Energy Efficiency:** Since embedded system has its own mobile battery which is not stable as electric plug at our home, we have to care about energy efficiency. We plan two things. The first one is wake up on LAN (WOL) technique. That turns off main computation device when there is no LAN connection and working as just normal headphone. If there is LAN connection, it wakes up and work as our Music Hub. The second thing is data compression. If the amount of data we send is reduced then the energy we are spending also would decrease.

**Security:** Mobile devices are exposed to public more than static machine because it may connect to various APs. To reduce the hazard involved by this expose, we are considering to make pass motion. It is like password or fingerprint but use user’s actual motion as password. It increases security more on top of previous methods.

### 6. Conclusion

In this paper, we present a prototype for a novel way of music sharing. As a main feature, it plays sync with other devices, a bunch of speakers installed in different devices can work as a single speaker. With several constraint play based on region or client, we can make various fun scenarios in our life.

As a one possible extension in future, we plan virtual sound channel (VSC). The name seems quite stale though, the concept is totally new. As we live in physical sound channel that we can hear voice or sound via air vibration, we can make another channel convey sound via networking. While physical channel has physical obstacle like wall, VSC based on network partially overcome the obstacle and make totally different experience.

It is just a prototype though, we expect our trial can change our life in future.