SALTApp : Sound And LighT Application
Speakers and stobe light with multiple smartphones

Alexandre Allani
School of computing
KAIST, Korea
Email: alexandre.allani@telecom-bretagne.eu

Faycal Baki
School of computing
KAIST, Korea
Email: faycal.baki@telecom-bretagne.eu

***Abstract*—Today, it is possible to have really good sound displayed on bluetooth speaker. However there are a lot of situation where one doesn't actually have a bluetooth speaker on him. Moreover, those devices need to be charged and are for some of them quite cumbersome. Moreover, inside a house, where bluetooth speakers might be usable, it is quite difficult to have a nightclub atmosphere, since strobe lights are quite expensive and inadapted to homes**
In this paper, we're going to discuss about a system that uses two (or more) smartphone to create a general speaker, by synchronizing the speakers of each smartphone, and a strobe light, using torch light of each smartphones. First we developped an application that switched on and off torch light of one smartphone according to the beat of the music, and then we developped an application that shares a music between two smartphones, plays the track and turn the torch light into a strobe light.

Keywords

Smartphones, Bluetooth, Beat Detection

1. Introduction

Today, there are different way to have portable music. Using bluetooth speakers is the privileged way. However, they might be sometime quite expansive, and they still need some charging. Furtheremore, it is also difficult to have what we qualify as nightclub atmosphere in this paper, at home or outdoors. Indeed, strobe lights are usually expensive and not only they need some knowledge to manage them, but also they are not transportable.That's why we propose an application, that could use only smartphones to do both be speakers and strobe light. Usually, when a user want a nightclub atmosphere, he is with some people. Hence, the user with his group, without more devices, can experience this atmosphere, just by using the SALTApp.

Basically, when entering the application, the user can choose a music to send. The he enters in a room and wait for other smartphones to connect to the room (ie wait for other users to connect to his smartphone by bluetooth). Then he just have to press the button send, and everything is done simultaneously. The music will be played, and lights switched on and off according to the rythm of the music. The purpose of the system is to give the user just a sense of a nightclub atmosphere and is not meant to perfectly imitate them.

We faced some challenges while creating this application. Our first challenge was to made an accurate beat tracker. We thought of different solution :

- Naive method : given a certain amount of time (a chunk), compute the instant Energy and compare it to the average energy of the chunk. If the difference between those two are greater than a thershold, there is a beat. This method is quite easy to implement and does not take a lot of computation time. However, it is really innacurate, and relies heavily on the chosen threshold and size of the chunk [1].
- Advanced method : Compute the spectrogram of the music. For each bin of time of the spectrogram, compute the average. Then compute the differences of average between two consecutive bin of time. The resulting array is called the spectral Flux, and gives us basically a curve describing how quicly the power spectrum of the signal is changing. With an adapted threshold we can get the beat of the music [2]. The drawback of this method is the computation time.

Eventually, we took the second method. Synchornizing the sound between smartphone, and the lights with the music is still a challenge up to this point. We succeeded to synchronize the light and sound, however it is highly dependent on the smartphone used. We will discuss in **Section 3** the issues of synchronisation.
Conerning the design, we had 2 main requirements we wanted to respect :

- The system must not rely on Internet connection. We want our system to be transportable and used in every possible situation. Using Internet would have been easier for synchronization, but would not have respected this characteristics.

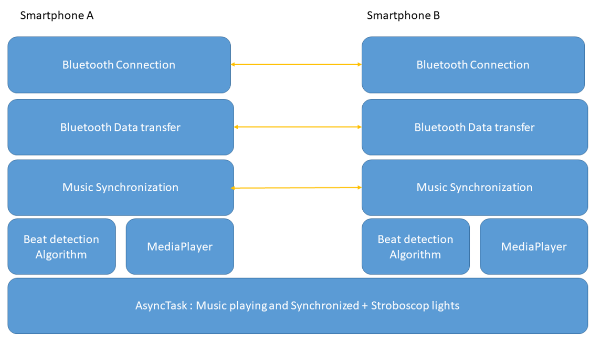

Figure 1. Architecture of the system

- The system must work with every smartphone, even if they don't have the music in them. Indeed others smartphone are considered as speaker, hence they are not supposed to have the music which will be played.

We will first present our contributions and related applications in section 2, then we will describe an overview of the system in section 3. In section 4, we will go in depth and explain how th system actually works. In section 5, we will review our implementation, and finally in section 6 how we evaluate our system. mds
June 18, 2018

2. Contribution and Related applications

Related application :
Ampme: This application uses other smartphone in a cluster as speakers. This application uses Internet to synchronize both smartphones.
Music Strobe: With ambiant sound captured through the microphone, transform the torch light into a strobe light. This applicaiton however works only with the microphone, and can sometime be too coarse grained.
Our main contributions are :

- We created an application that can track beat in a music and switch on and off the torch light of the smartphone according to those beats.
- We created an applcation that turn smartphones in both speakers and strobelight.
- The system can be used offline, and needs only bluetooth

3. System Overview

Our app enables two Android smartphones to connect through Bluetooth, one of them as a host and the other one as a client. When connected, the host smartphone can decide to start playing a song. For now, the user cannot chose the song. The app works with a single song, which is stored on the root of the device storage. After being downloaded by the client smartphone through Bluetooth, the song will start playing on the two smartphones at the same time. While the song is being played by the two smartphones, each smartphones flashlight will work as a strobe light, blinking in rhythm with the music.
Our app has 4 functionalities. On the launch screen, the user has two choices :

- **Host a party** : This will make the device discoverable by Bluetooth, allowing one other device with the app to connect to it.
- **Join a party** : This allows the device to enable its Bluetooth to later connect to a host.
- If the user has chosen to join a party, they can then use the following functionality: **Connect** : This allows the device that has chosen to join a party to choose which host device to connect to.
- If the smartphone is hosting a party and is connected, the user can use the 4th functionality: **Play a song** : The play button allows the host smartphone to start transferring the song to the client. When the transfer is done, the two smartphones start playing the song at the same time.

As the architecture on **Figure 1** describes it, there are 5 layers. The application is assymetric as far as the functionalities are concerned. But as far as the architecture is concerned, since each smartphone should be able to be a sender or a receiver the architecture is the same for smartphone A and B. The architecture is divided in 5 layers :

- The first layer just initializes the bluetooth connection between the two smartphone. While the connection is not established, the other steps cannot be done. This means that for the moment, if you are alone, you can't use your smartphone as a "unique" strobe light.
- The second layer will handle the transfer of music through the Bluetooth connection. A special protocol has been created so that the communication between the two smartphone are c.
- When the application reaches the third level, it will engage the Synchronization procedure. This procedure is based on the time taken by a bluetooth request to do one round trip.
- On the fourth level, the Android MediaPlayer that allows us to play a music will be initialized. In the same time, the beat detector Algorithm will compute an array containing all the beats of the music and their time associated.
- On the fifth level, an Asynchronous Task will play the music and switch on and off the torch light on each smartphone according to the array created on layer 4. Thanks to layer 3, the music should be played asynchronously

The next section will describe how the main parts of the system works

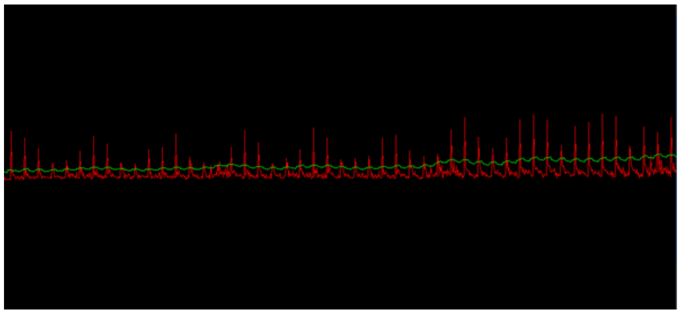

Figure 2. Result of the Beat Detection Algorithm. In red, the variation of power given by the spectral Flux, and in green the threshold curve.

4. SALTApp

The application is divided into two distinct part. One is the tracking of beats and how to associate beat detction to torch light in order to have a strobe light effect. The other one is the data transfer through bluetooth, and music synchronisation.

4.1. Beat Detection Algorithm

4.1.1. Spectral Flux

. As we said in the Introduction, a lot of methods exist to track beats in a music [1,2]. But we chose one based on the Spectral Flux [3,4]. First off, we can modelize a music by its spectrogram. The spectrogram is a table where the columns correspond to time and the rows to frequency. The values inside the cells are the amplitude corresponding to a couple (Frequency, Time).
With this spectrogram we can calculate the Spectral Flux : for each bin of time, we compute the difference of the current spectrum, and the previous one.

$$SF(k) = \sum_{i=0}^{n-1} s(k,i) - s(k-1,i)$$

The spectral flux SF is a mean to measure the overall variation of power between two bin of time. The values inside the spectrogram gives the amplitude for each frequency, calculating the spectral Flux, is just seeing the variation of the mean amplitude. You can see the spectral Flux of one song on **Figure 2** (Red curve). As we can see, we have distinct peaks on the Spectral Flux curve. However we still need to determine their position precisely.

4.1.2. Threshold function

. Now we have the spectral flux, we should be able to determine the peaks in the music. However, according to the threshold, we can take only the highest peak (ie big drums), or we could take more. Having a constant threshold is a bad idea since the same variation of power could be huge for one music and nothing for another.
For instance, in a music where there is silence, a note is played and then silence again, the variation of power would be clearly seen in the spectral flux. However, in a real music, the same note played with the same intensity could be as low as the ”noise”. Therefore, to calculate the threshold we derived the spectral flux [4] with the finite derivative formulat

$$TH(k) = \frac{SF(k) - SF(k-7)}{7}$$

The value of 7 has been chosen after some testing over 4 different techno-typed music. Basically, the higher this value is, the more smooth the derivative is. Hence, if the value was equal to 20, the threshold function would be almost constant, and this is not wanted.
Then, we multiplied this derivative function by 1.9. Indeed otherwise, the derivative function would be too low and would cross the Spectral Flux curve when it is noise. The value of 1.9, has also been chosen according to tests over 4 different techno-typed music. With all of that, we have our threshold function.
With the threshold function and the spectral flux, we consider that there is a peak when the threshold curve the spectral flux one. With this we obtain an array with all the peaks.

4.1.3. Creation of strobelight application.

Having an array detecting the peaks, we created an application that, with a given music and our beat detection algorithm, switched on and off the torch light of a smartphone. This was a challenge, because of the synchronization between the light and the sound. With the previous subsubsection, we have an array describing the moment when a light should appear.
To synchronize both light and music we used an Asynchronous task in android. With a MediaPlayer we start playing the music. Knowing that the samplerate used is 44100 KHz, and that the windows resolution for the spectrogram used was 1024, we have a time resolution of 23 ms. The naive way to synchronize light and sound would be to put a timer and each 23 ms switch on and off the light according to the light array. However, this would not take into account the time taken by the application to execute the lines that turns on and off the light.
In order to solve this issue, we created a dynamic timer [5]: each time a light should be turned on or off, we compute the time taken by the algorithm to run, and substract this value to 23ms. This give us the real time to wait.
Finally, we also had to set the highest priority to our thread, otherwise, Android system would sometime leave the thread for more than 100 ms, to execute another thread. This was of course unacceptable since we had a resolution frequency of 23ms.

4.2. Bluetooth and Data Transfer

We designed then implemented a simple Bluetooth protocol for the song transfer. All the Bluetooth transfer is done through 1024 bytes byte arrays. The app does not allow streaming yet. Instead, a song is entirely downloaded by the client device before being played.
When the host devices to start playing a song, a first 1 kB Bluetooth packet is first sent to the client device to tell it that

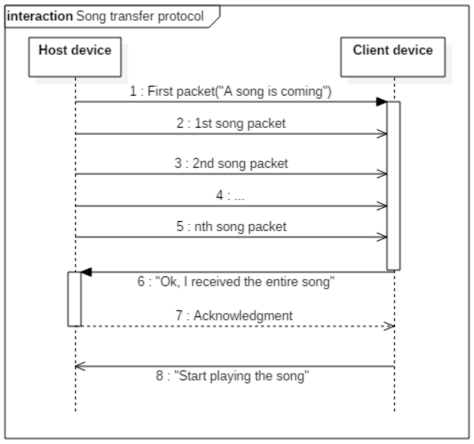
interaction Song transfer protocol
Host device
Client device
1 : First packet("A song is coming")
2 : 1st song packet
3 : 2nd song packet
4 : ...
5 : nth song packet
6 : "Ok, I received the entire song"
7 : Acknowledgment
8 : "Start playing the song"

Figure 3. Protocol of th bluetooth data transfer.

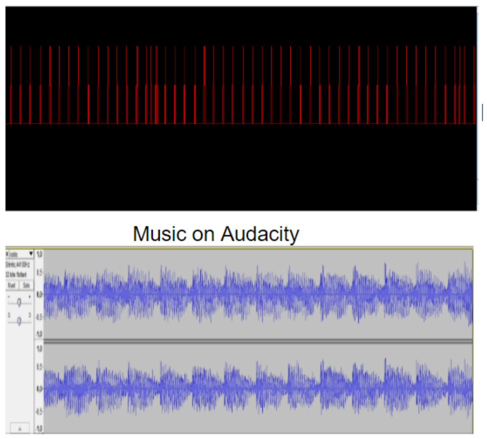

Figure 4. Evaluation of the light part

a song is coming. The first 8 bytes of this packet are 0, 1, 2, 3, 4, 5, 6, and 7. This sequence allows the client device to know that a song is incoming. The 9th, 10th, and 11th byte of this array are used to store the size of the song in kB. The maximum size allowed is then 127*127*127=2,048,383 kB, which is more than 2 GB, and enough for any song. This way, the client device knows exactly how many packets it has to receive before to start playing the song.
Then, the host sends the entire song in 1024 kB packet. In the meantime, the client devices counts how many packets are received.
When the client has received the expected number of packets, it launches the strobe light algorithm to compute when to turn the light on and off. When this is done, it sends a packet back to the server device, telling that it has received all the song. Then, the server sends an acknowledgment packet back to the server. The client measures the time difference between the last two transfers. Lets call this time difference T.
Finally, the client sends a packet to the server device, ordering to start playing the song. It then waits for T/2 before starting playing the song. The service devices starts playing the song instantly when it has received this packet.Note that even though the app works with only one song for now, this protocol can be used to play multiple songs in a row.

5. Implementation

We implemented the applications on two smartphone Samsung galaxy S7 edge, and one smartphone Samsung galaxy S8 +. We use the Bluetooth API to do the bluetooth part. As far as the light part is concerned, we used the CameraManager API for the light, and the MediaPlayerManager API to play music. We used an external library to read wavFiles [6].

6. Evaluation

We tried to evaluate our system through different test. First, let's talks about the light part. For this part, it was difficult to know what the ground truth was. So we used Audacity as you can see on **Figure 4**. Thanks to the beat detection algorithm, we know when the light is suppose to switch on and off. We then compare the peak in the musics and when a beat is suppose to happen. This method of evaluation is quite inneficient and take a lot of time.
Moreover, for this paper we only took techno music, because of the bass. Indeed, the beat detector algorithm track much better song with heavy basses than songs without. This is due to the fact that our threshold is the derivative of the variaiton of the power spectrum. Because of that, a music without a lot of bass, will have a threshold higher than the power spectrum on the average. This is why we took only techno music.
As far as the bluetooth part is concerned, we ran a lot of tests with song and text file to see if our protocol worked. In the begining we had a lot of trouble making it working, but in the end, the music sent did not have missing bytes. For the synchronisation part, we tried to see with the functions System.nano the time taken by each thread. However, the timer in Android system is quite inefficient and we did not take into account some part of the code that could have been executed by other threads. We need to do a better evaluation of synchronisation.

7. Conclusion

We presented SALT, a prototype for a music streaming app, using only Bluetooth. The main contributions are the fact that our app does not require any external server or hotspot, and the strobe light effect that works in rhythm with the music.
This app lacks speed and reactivity for now, because it needs a song to be completely downloaded by the client before being played.
In a future work, we would focus on allowing the devices to play songs while downloading them, by buffering chunks of audio, and allowing them to switch between songs seamlessly.

Acknowledgments

The authors would like to thank the authors of [1] and [3], that really helped us make a working beat tracking algorithm.

References

[1] http://mziccard.me/2015/05/28/beats-detection-algorithms-1/

[2] https://www.badlogicgames.com/wordpress/?category_name=onset-detection-tutorial

[3] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.332.989&rep=rep1&type=pdf

[4] https://www.badlogicgames.com/wordpress/?category_name=onset-detection-tutorial

[5] https://www.gamasutra.com/view/feature/171774/getting_high_precision_timing_on_.php?page=3

[6] http://www.labbookpages.co.uk/audio/javaWavFiles.html#methods