Real-Time Routing Protocol for Music Application With Multiple Outputs

Jos Timanta Tarigan, Herriyance, Kris Pradana Washitha Nala
Faculty of Computer Science and Information Technology, Universitas Sumatera Utara
jostarigan@usu.ac.id

Abstract. We present a client-server based smartphone application that act as a music input device with several outputs. The application allows user to interact with one phone as the input and the server and hear the sound from several connected smartphones that act as clients. Since transmitting data may require some time, we need to deal with transmission delay. Hence, we developed an algorithm to detect the delay and synchronize these values between all connected devices. By agreeing to a single delay value, we were able to minimize the difference in playtime between server and connected clients. We performed the test by building an Android-based smartphone application, installing it to multiple devices, and connecting them to a WiFi-based local area network. The test shows that our proposed system has successfully reduce the difference between devices and the connected devices were able to produce sounds with insignificant delay.

1. Introduction
The development of computer network technology allows multiple device to work collaboratively to achieve a better result. With the currently available technology, streaming massive data can be done faster with a minimum delay. Hence, there are various research focus on developing a network-based multimedia system such as distributed system, collaborative work, and cloud-based system.

In this research, we develop an application that allows a device to control other connected devices to act as it’s sound output. The main device, called master, will act as a music device and others connected devices act as the output speaker. Additionally, to sync the sound generated by the speakers, we developed an algorithm to calculate the delay between devices and distributed the average delay time to the connected devices. This method is intended to reduce the difference of play time between devices and, in result, produce a more synced sound.

In this paper, we will go through the proposed solution. The paper is structured as follows; in the next session, we will go through a few related researches that has been published earlier. In the third section, we will go through the design of our system and the protocol used in the system. In the fourth section, we will discuss the developed application that used the proposed design. We will also present the test data. We will present our conclusion and future projects in the last section.

2. Related Works
There are various previous researches that focus on developing a distributed multimedia system. Ning Li developed a distributed audio system based on ANSAware which can be run on different platform [1]. Riedl et al. developed SuiteSound, a support system that intended to manipulate multimedia flows,
such as digital audio, that is uniform and independent of any physical device [2]. Another interesting project related to the topic is presented by Sergi et al. who developed a collaborative music composition over the web [3]. The developed system allows a real-time jamming session by multiple authors over the internet. In this project, they used Transjam protocol developed by Philip Burk that specifically built to distribute sound between multiple musician [4] and has been used by multiple research projects such as Auracle [5] and the FMOL Project [6].

Another important topic related to this research is the delay synchronization protocols. There are multiple methods available with various platform, performance, and media. Blakowski et al. and Boronat et al. performed a comparison study of available methods [7], [8]. Ravindran et al. presented a delay compensation protocol for multimedia systems that connected through a broadband ISDN’s and metropolitan area networks [9].

3. System Design
In this section, we present the design of the proposed system. We will go through the overall architecture of the system and briefly describe the protocol that is applied to the system. As previously stated, the objective of this project is to implement a music device with multiple outputs. We do this by using an application installed in multiple smartphone that connected and play a synchronized note. These devices will be connected in a local area network, preferably through a wireless network. To ease the implementation, we will use smartphone as our platform.

Figure 1 below shows the overall architecture of the application and the protocol used to synchronize playtime between devices. As can be seen in the diagram below, there are two roles that can be assigned to the connected devices: master and slave. The master is responsible for the input device. In our case, the input is a set of keys which each is assigned to a single note. The other devices will be assigned as a slave; its responsibility is to play the note pressed by the master.

![Figure 1. The architecture of our distributed music devices system](image)

To synchronize the playtime, we developed a communication protocol between master and slaves as shown in Figure 2. The playtime sync protocol is as follows: when a button is pressed, the master device will send a request to play a note followed by the default button pressed time 0 and the default delay time 0. Slaves will then play the note and send the playtime back to the master device. These values will be averaged and set as the new delay time. The new delay time will be sent along with the pressed time. If the command received by the slave before the delay time expired, the slave will wait until the delay time is expired and play the note. However, if the command is received after the delay time is expired, the slave device will play the note immediately. After the note is played, the slave will send the new delay time and the process is repeated all over again.
4. Implementation

To test the proposed solution, we build an Android smartphone application based on the previous design called Orkestra. Figure 2 below is the user interface of our system. The main interface (middle) consists of a set of squares that represents a note. The user pressed one of these squares to play a note. The current version of the application has 3 types of sound: piano, guitar, and trumpet. We use waveform audio format (.WAV) as the audio file format.

The connection setup has to be performed manually; users have to connect the master device to the router and check the IP address assigned to the device. To set a connection to the master device, the user can enter the IP address of the device that set as the master. If the device is the master, the user can enter the localhost IP address (127.0.0.1). The system uses port 4444.

![Figure 2. The Playtime Synchronization Protocol](image)

In our first test, we used 1 device as a master and 4 devices as slaves. All of this devices were connected through a standard 2.4 WiFi router. Additionally, all devices system time have been synchronized. The table below shows the statistic of delay times gathered from the slaves and the average of the delay. The most-left column is the test note sequence in one session. The second and third columns show the minimum and maximum delay time gathered from the 4 slave devices. The fourth column is the delay time calculated from the average delay time gathered from the device.

| Note Sequence | Min Delay (ms) | Max Delay (ms) | Average Delay (ms) |
|---------------|----------------|----------------|-------------------|
| Piano         | 100            | 200            | 150               |
| Guitar        | 150            | 250            | 200               |
| Trumpet       | 200            | 300            | 250               |

![Figure 3. The GUI of Orkestra; splashscreen (left), main interface (middle), and server setup (right)](image)
Table 1. Test Result for 3 Devices

| No. | Min (ms) | Max (ms) | Delay (ms) | Avg Playtime Diff (ms) | Expired Delay (num) |
|-----|----------|----------|------------|------------------------|---------------------|
| 1   | 77       | 161      | 0          | 112                    | 4                   |
| 2   | 13       | 154      | 61         | 39                     | 1                   |
| 3   | 30       | 159      | 55         | 51                     | 1                   |
| 4   | 45       | 184      | 66         | 76                     | 2                   |
| 5   | 16       | 151      | 49         | 41                     | 1                   |
| 6   | 24       | 123      | 57         | 50                     | 1                   |

Since there is no previous data on the first attempt, the delay time is set to 0 millisecond. Consequently, upon receiving the command, all slaves will acknowledged it as an expired one and play the note immediately. Since 0 delay time will also applied to the master, the playtime difference is high. In the second and next attempt, we notice that there are at least 1-2 devices with expired delay. However, it is important to notice that the delay is reduced significantly. In the second test, we use more devices as the output. We add another device and the result of our test can be shown in table 2 below.

Table 2. Test Result for 5 Devices

| No. | Min (ms) | Max (ms) | Delay (ms) | Avg Playtime Diff (ms) | Expired Delay (num) |
|-----|----------|----------|------------|------------------------|---------------------|
| 1   | 63       | 192      | 0          | 133                    | 5                   |
| 2   | 15       | 123      | 51         | 76                     | 2                   |
| 3   | 21       | 114      | 67         | 41                     | 2                   |
| 4   | 34       | 98       | 76         | 31                     | 2                   |
| 5   | 46       | 151      | 56         | 43                     | 1                   |
| 6   | 41       | 152      | 62         | 35                     | 1                   |

However, based on the values we get from the test, the value does not converge to zero. More data does not necessarily increase the synchronization performance. The minimum and maximum average playtime difference occurs randomly and there is no pattern.

5. Conclusion

In this paper, we present a distributed music application that allows a user to use one master device and distribute the output to multiple slave devices. The master will send the pressed note to the others to be played on those devices. To decrease the playtime delay between devices, we developed a protocol to calculate the average delay between the devices. Based on the result we gathered from the test, the protocol is capable to reduce the delay time after 2 or 3 notes. However, playing more notes will not reduce the delay further. There are also cases when the delay is increased due to the spike of maximum delay. We can conclude that while the protocol may reduce the playtime delay, the effectiveness of this method depends on the quality of the network.

While our system is able to simulate distributed music devices with multiple output, there are a few improvements that can be made to the future. Synchronization is still an issue in our proposed system and there are a few previous researches that focus on synchronizing devices using round trip time that can be implemented in our system.

Furthermore, this project is a kick-start of our digital orchestra project using distributed systems. Our intention in the future is to add more types of sound and add various music devices to the application.
References

[1] N. Li, “A Distributed Audio System,” in Multimedia/Hypermedia in Open Distributed Environments, W. Herzner and F. Kappe, Eds. Vienna: Springer Vienna, 1994, pp. 109–121.

[2] J. Riedl, V. Mashayekhi, J. Schnepf, M. Claypool, and D. Frankowski, “SuiteSound: a system for distributed collaborative multimedia,” IEEE Trans. Knowl. Data Eng., vol. 5, no. 4, pp. 600–610, Aug. 1993.

[3] S. Jordà and O. Wust, “A system for collaborative music composition over the web,” 2001, pp. 537–542.

[4] P. Burk, “Jammin’on the web-a new client/server architecture for multi-user musical performance,” Jan. 2000.

[5] C. Ramakrishnan, J. Freeman, and K. Varnik, “The architecture of auracle: a real-time, distributed, collaborative instrument,” Jan. 2004.

[6] S. Jordà and Á. Barbosa, “Computer supported cooperative music: overview of research work and projects at the Audio Visual Institute–UPF,” Dec. 2019.

[7] F. Boronat, J. Lloret, and M. García, “Multimedia group and inter-stream synchronization techniques: A comparative study,” Inf. Syst., vol. 34, no. 1, pp. 108–131, Mar. 2009.

[8] G. Blakowski and R. Steinmetz, “A media synchronization survey: reference model, specification, and case studies,” IEEE J. Sel. Areas Commun., vol. 14, no. 1, pp. 5–35, Jan. 1996.

[9] K. Ravindran and V. Bansal, “Delay compensation protocols for synchronization of multimedia data streams,” IEEE Trans. Knowl. Data Eng., vol. 5, no. 4, pp. 574–589, Aug. 1993.