Development And Testing Of a Bluetooth Multi-Room System

Michael Jensen, mj09@student.aau.dk
Computer Science, Aalborg University

June 2018

Abstract

In this paper we implement a Bluetooth audio streaming multi-room system which we proposed in [5]. This system is then tested both in piconet and scatternet configurations. This is accomplished by streaming audio from Android devices with different versions from 4.0.3-8.0 and different Bluetooth versions from 2.1-4.2. The results of these tests showed that Bluetooth can be used to create such a system, but with certain downsides. Meaning that it does not support audio streaming at CD quality, but has to degrade the audio by lowering the sample rate, bit depth and stereo/mono settings of an audio file. By doing so showed it to be possible to audio stream to several devices in different configurations of a scatternet.

1 Introduction

A wireless multi-room audio streaming solution has become a very common household belonging in a modern home. Where brands like Sonos, Marshall, Bose, Denon and so forth have similar solutions to achieve wireless multi-room audio streaming. What these brands have in common is that they all use WiFi to connect their speakers together and receive audio from a smart device such as a smartphone or tablet.

In [5] we did several discoveries by testing audio streaming over Bluetooth. The testing methodology and which devices were tested can be viewed in the paper. We noticed that Bluetooth, regarding audio streaming, can have a larger range than advertised for the product.

Having a newer version of Bluetooth, at least between Bluetooth 3.0 and 4.0, seemed not to have any effect on the range of the connection. A Bluetooth 3.0 device outperformed a newer Bluetooth 4.0 device, meaning that the range of a Bluetooth connection comes down to the power class of the Bluetooth device [5].

Another discovery showed that it was possible to establish a working Bluetooth connection through obstacles such as concrete walls. However that depended heavily on which device was used. The devices with a strong power class did not lose any range by going through walls compared to a test with no obstacles. The measurements of these tests can be found in [5].

Furthermore our testing showed that Bluetooth seems to handle interference quite well which matches the findings in [3] where it showed that Bluetooth handles interference better than WiFi. The reason for this is that Bluetooth can hop to a less noisy channel if it experiences noise on the currently used channel. The same paper also gave insight into Bluetooth being 1.6-2.2 times more power efficient than WiFi when transferring with low throughput, such as audio streaming. Their findings showed that the power consumption of a phone barely increases when receiving audio over a Bluetooth connection compared to playing the audio locally.

From these findings we can deduce several discoveries regarding audio streaming:

- The range of a Bluetooth connection is heavily dependent on the Bluetooth power class and not the Bluetooth version of the device.
- A Bluetooth connection, with a strong enough

power class, has the capability to maintain a connection on at least 14.3 meter in a house by overcoming obstacles such as walls.

- Bluetooth handles interference better than WiFi. However in our testing the range was affected slightly when interference was introduced.
- Bluetooth is more power efficient than WiFi.

Given these discoveries we can develop a Bluetooth multi-room system and discover if it is feasible to audio stream to several devices as it is with one of the known WiFi multi-room solutions. If it is feasible and it performs equivalent to a WiFi solution then there is a strong case for applying Bluetooth technology into multi-room solutions because of its power efficiency.

1.1 Paper Organization

The rest of the paper is structured as follows; In Chapter 2 we investigate audio by looking at the different parameters of an audio file and examine different Bluetooth versions.
In Chapter 3 we explain our implementation of the Bluetooth multi-room system and the overall architecture. Chapter 4 explains our testing of the implemented system and afterwards in Chapter 5 we discuss the obtained results.
Lastly in Chapter 6 we give a conclusion on what has been achieved and in Chapter 7 we give our opinion on what could be interesting to investigate further.

2 Theory

In this chapter we will take a look at the different parameters of an audio file such as sample rate and bit depth. Besides that we will also look at the difference between stereo and mono and what Pulse-Code Modulation (PCM) is. This is necessary to understand before we start implementing our system and begin testing it. Lastly we will also take a look at Bluetooth again regarding the different versions and the Advanced Audio Distribution Profile (A2DP) which is used to stream audio to Bluetooth enabled speakers.

Sample Rate in Hz	Maximum Frequency in Hz
8000	3600
11025	5000
22050	10000
32000	14500
44000	20000
48000	21800

Table 1: Sample rates and the maximum frequency that can be achieved for each sample rate

2.1 Sample Rate

A sample, in the realm of music, is a sound recording of an instrument that combined with other samples can produce audio. Whereas Sample rate can be defined as the number of samples of audio carried per second which can be measured in Hz. The sample rate decides the maximum audio frequency that can be reproduced, this can be observed in Table 1.

44100 Hz is the sampling rate used on CDs which gives a maximum frequency of 20000 Hz. This is generally the highest frequency which is audible by the human ear. However the ability to hear frequencies at 20000 Hz swiftly falls with age. Meaning that people in their teenage years can hear 20000 Hz while many older people can not hear above 14500 Hz.
32000 Hz is suitable for cassette recordings, speech and all other audio where a smaller file size than 44100 Hz recordings are desired with only a little compromise in sound quality.
22050 Hz has been a popular choice for low bit rate MP3s. The audio quality is considerably affected, because higher frequency content is missing. It is suitable for speech where the quality is not important but clarity still remains, such as AM radio and to transfer Mp3 files onto small Mp3 players.
Going down to 11025 and 8000 Hz results in bad audio quality and should be avoided [1].

Bits	SNR
8	48.16 dB
16	96.33 dB
24	144.49 dB

Table 2: Three different bit depth and their SNR

2.2 Pulse-Code Modulation

PCM is a method that is used to digitally represent sampled analog signals. This is used as the standard form of digital audio in computers. A PCM signal is an array of digital audio samples that provides the data with the necessary information to rebuild an original analog signal, e.g. audio [9].

2.3 Bit Depth

Bit depth in digital audio with PCM is the number of bits of information in each sample and it can directly be compared to the resolution of each sample. a CD uses 16 bits per sample and a DVD can support up to 24 bits per sample. Having a higher bit depth results in a higher signal-to-noise ratio (SNR). For example a CD has a SNR of 96.33 dB, which means that the level of the audio signal is 96.33 dB higher than the level of noise. Observing Table 2 shows the bit depth of 8 and 24 bit as well and their SNR [8].

2.4 Stereo and Mono

The difference between stereo and mono is the number of channels used, e.g. signals. Mono uses one while stereo uses more than one.
Using mono sound the single channel can be reproduced through several speakers, but all the speakers produce the same copy of the signal.
Applying stereo sound makes is possible to use two different channels and make one feed one speaker and the second channel feed another speaker. Table 3 illustrates several sound configurations with different sample rate, bit depth and stereo/mono, and the amount of bytes needed per second to sample audio at each configuration [6].

2.5 Bluetooth

Bluetooth was developed in 1994 and has since been launched in several updated versions. The different versions can be observed in Table 4. Regarding Bluetooth 3.0 and 4.x with a transmission rate of 24 Mbps is only if High Speed (HS) is enabled, if not then its transmission rate is 3 Mbps. Bluetooth 5.0 promises double the transmission rate of basic rate (BR), Enhanced Data Rate (EDR), HS and, Low Energy (LE) Bluetooth. The transmission rate of LE is between 125 Kbps to 2 Mbps. The max payload size of LE is 251 byte while BR and EDR is 1021 byte.
Bluetooth versions are backwards compatible. Meaning a Bluetooth 4.2 device can easily connect to a Bluetooth 2.1 device, however the active connection will work as a Bluetooth 2.1 version then. Obviously LE connections can only be established from Bluetooth version 4.0 and above.
Regarding the HS protocol for it to work the Bluetooth chip it self has to support it. Furthermore transferring data is not accomplished over the Bluetooth link it self, but instead over a 802.11 link. The Bluetooth link is only used to negotiate and establish the connection.

When connecting a phone over Bluetooth to a speaker, car, headset and so forth with the intention to stream audio then this is accomplished by the A2DP profile. This profile makes it possible to stream any audio that is being played on the device itself to a connected Bluetooth device. It supports two different qualities called Middle Quality and High Quality where these support both mono and stereo. The sampling frequency that can be used is either 44100 Hz or 44800 Hz in several configurations. The bit rate to sample audio at the lowest quality is 15875 bytes per second and the highest is at 43125 bytes per second [7].

3 Implementation

In this chapter we will discuss which technologies we used to develop the system, which implications there exist and how we can overcome them. Next we will take a look at how the overall architecture of the

Sample Rate in Hz Bit Depth Stereo	48000 16 Bit	44100 16 Bit	32000 16 Bit	24000 16 Bit	22050 bit	16000 16 Bit	11025 16 Bit	8000 16 Bit
Bytes Per Second	192000	176400	128000	96000	88200	64000	44100	32000
Sample Rate in Hz Bit Depth Stereo	48000 8 Bit	44100 8 Bit	32000 8 Bit	24000 8 Bit	22050 8 bit	16000 8 Bit	11025 8 Bit	8000 8 Bit
Bytes Per Second	96000	88200	64000	48000	44100	32000	22050	16000
Sample Rate in Hz Bit Depth Mono	48000 16 Bit	44100 16 Bit	32000 16 Bit	24000 16 Bit	22050 16 bit	16000 16 Bit	11025 16 Bit	8000 16 Bit
Bytes Per Second	96000	88200	64000	48000	44100	32000	22050	16000
Sample Rate in Hz Bit Depth Mono	48000 8 Bit	44100 8 Bit	32000 8 Bit	24000 8 Bit	22050 8 bit	16000 8 Bit	11025 8 Bit	8000 8 Bit
Bytes Per Second	48000	44100	32000	24000	22050	16000	11025	8000

Table 3: Bytes per second needed to sample audio for each sound configuration

Bluetooth Version	Features	Data Rate In Theory	Range In Theory
1.2	Basic rate	1 Mbps	10m
2.1	Pairing EDR	3 Mbps	30m
3.0	HS	24 Mbps	30m
4.0 4.1 4.2	LE	24 Mbps	60m
5.0	IoT	48 Mbps	200m

Table 4: Table showing different Bluetooth versions. Every newer version also has the same features included as the versions before it. EDR stands for Enhanced Data Rate, HS is for High Speed, LE is for Low Energy and IoT is for Internet of Things

system will be implemented. Lastly we will explain the implementation, show relevant code and highlight specific complications and how they were solved.

3.1 Development Process

The most obvious way to develop a Bluetooth multi-room system would be to develop an application for a smartphone and stream audio to multiple Bluetooth speakers. However such an approach has it limitations. Firstly, the only way to connect to such speakers is by using the A2DP which Android and IOS only support one active connection at a time [4].
Secondly, even if there was support for more active connections with A2DP one would also run into the limit of how many connections a Bluetooth device can have active at a time. As explained in [5] a Bluetooth device can create a piconet between it self and seven other active connections. Furthermore it can be expanded into a scatternet which consists of two or more piconets where each piconet have a master. A master can also act as a slave in different piconets. Naturally Bluetooth speakers, except some UE Boom speakers, do not have the ability to create a scatternet like this. However the UE Boom speakers with this capability are not designed as a multi-room solution and they only work by UE Boom's own application. This means we can not control how the scatternet is deployed and even more important control the audio stream between the devices.
Thus we need to develop a system from the ground up to be able to have a fully functional Bluetooth multi-room solution. To accomplish this and to have full control over the scatternet, and each master and slave of the piconets, we will develop an application for the Android operating system.

3.2 Architecture

The architecture of the system will be a client-server architecture with one true master smartphone, the streaming device, clients that will play the streamed audio and master clients that can both act as a client and play audio or it can stream the audio further to clients in their own piconet. An example of the architecture is illustrated in Figure 1. The reason for

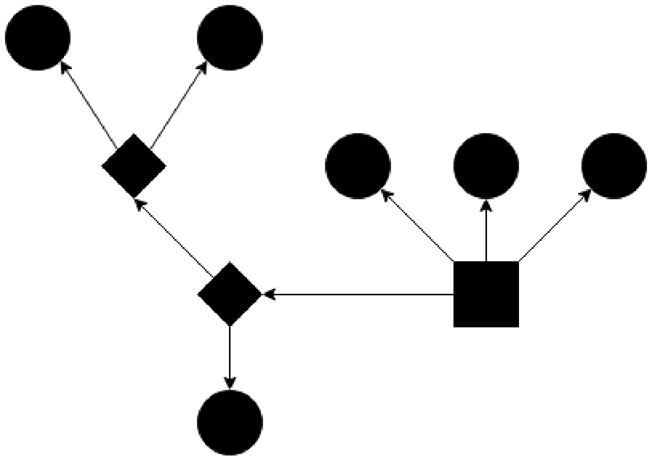
Figure 1: Illustration of the architecture of the system. The squared figure is the master device, the diamond figures are the master client devices and the circular figures are the clients.

creating a scatternet like this is twofold. Firstly, as pointed out earlier, a maximum of eighth Bluetooth devices can be in one piconet. Secondly, the range of the master device can be greatly increased by use of a scatternet.
The communication between each device is limited to a one way communication model, meaning that the master of each piconet streams audio to its clients. The majority of Bluetooth chips have a default value of retransmission timeout (RTO) as infinite, which makes it reliable because it does not drop any packets [2]. Therefore, in our case, there is no need for the clients to confirm to the master that they have received the audio stream. However this can lead to a delayed stream and affect the audio quality in a negative way if the connection between the devices is poor, e.g. resulting in chopping audio.
Another possible issue here is synchronization between the devices, e.g. the devices are not playing the audio synchronized. However the goal of this project is not synchronization, thus if the devices do not play the audio synchronized it does not impact the final conclusion of this project.

Device	Android Version	Bluetooth Version
Huawei P10	8.0	4.2
Samsung Galaxy Tab S	6.0.1	4.0
Samsung GT-I9100	4.1.2	3.0
Samsung GT-I8190N	4.1.2	4.0
Samsung Nexus S	4.4.4	2.1
Huawei U8666E	4.0.3	2.1
Lenovo A3500-F	4.4.2	4.0
Lenovo TB3-710F	5.0.1	4.0

Table 5: Devices used in the Bluetooth multi-room system and their Android and Bluetooth versions

3.3 Devices

The devices that will be used to create the Bluetooth multi-room system can be observed in Table 5. One can note the different versions of Android on the devices in the table. Having such older devices may potential create issues or challenges because of not having access to the newest methods that are implemented on the Android platform. Furthermore the older devices may also limit the performance of the system because of obsolete hardware such as speakers, Bluetooth chips, CPU power and so forth. However we will explore this further in Chapter 3.5 and in Chapter 4.

3.4 Main Challenges

Implementing a multi room audio streaming system has several challenges, such as:

Challenge 1 Controlling the buffer that receives data from the sound file. Meaning that it should not read new data in before the current data has been sent to all connected devices. This is a necessity to ensure each connected device receives all of the sound file.

Challenge 2 Finding the most suitable buffer size to stream audio, with the intention to avoid buffering and chopping sound on each connected device.

Challenge 3 Discover if there exists any hardware limitations of the used devices with regards to streaming continuously to several devices. This is necessary to establish the maximum size of each piconet which is required to create the over all architecture of the system.

3.5 Code Review

Since we are using the devices from Table 5 and programming for the Android platform we chose to use Java as our programming language. We chose to use Android Studio as our IDE since it is developed for developing Android applications.
Furthermore Android only supports classic Bluetooth and LE Bluetooth, but not HS Bluetooth. But since many of our devices do not have LE Bluetooth we will be using classic Bluetooth.

3.5.1 System Design

Figure 2 shows the architecture of the system. The figure also illustrates the flow of the system depending on which activity has been chosen after the *MainScreen* class. As explained in Section 3.2 and visualized in Figure 1 the system is designed for a client server architecture with a master device, master client devices and client devices. From now on we will refer to these three as a master, master slave and slave device.

3.5.2 System Flow

In this section we will describe how the system works for our master, master slave and slave device. We will proceed with describing the device activities, e.g. *MasterActivity*, *MasterSlaveActivity* and *SlaveActivity*, and explain the flow throughout the system.

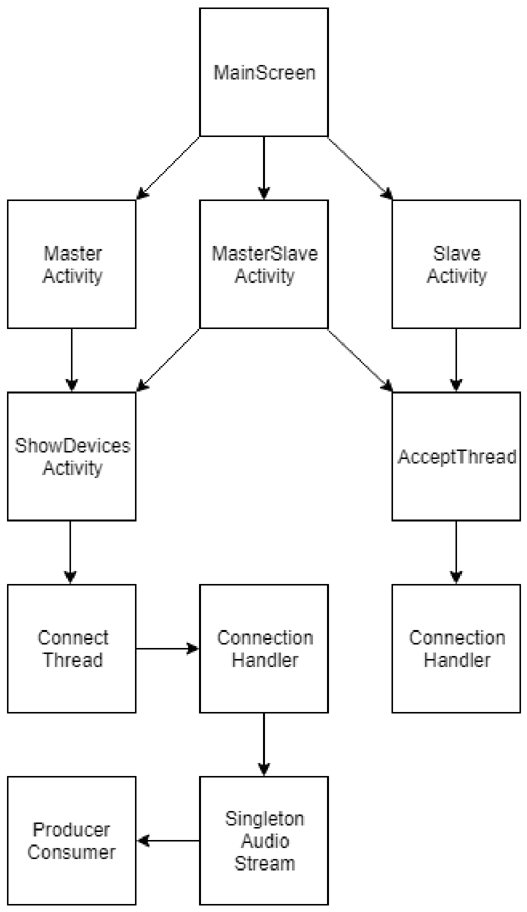

Figure 2: An overview of the system design and flow. Each square represents an activity/class.

Most of the explaining will be at a higher level of abstraction and only the more interesting implementation will be explained more carefully.

3.5.2.1 Master Device

We will begin by explaining the *MasterActivity* class and its flow through the system. In this class we have two different methods for loading an audio file. First we have a method where we create a new file from the chosen audio files file path.
This file will be sequentially read in real time in the *ProducerConsumer* class. Second method is that we preload the entire audio file into a buffer. The reason for having two different methods is to see if there will be any degradation in performance since the first method requires an extra I/O operation.
After that the class will begin to search for discoverable Bluetooth devices. This is accomplised by using the method *onReceive()* from the *BroadCastReceiver* class which will register any Bluetooth devices it locates and will also tell us the found devices' MAC addresses and device names.
Proceeding from the *MasterActivity* class to the *ShowDevicesActivity* class yields in a list containing devices that were found from the *onReceive()* method. By selecting one of the shown devices, moves us to the *ConnectThread* class. In this class we use the selected device's MAC address and a preset Immutable Universally Unique Identifier (UUID), which makes it possible to connect to the selected device.
When the connection is established we move to the *ConnectionHandler* class which handles the output to the connected device. This class starts the *SingletonAudioStream* class. This class is constructed in such a way that it only loads the audio file once no matter how many new connections are activated. Furthermore it also only starts one instance of the *produce()* method from the *ProducerConsumer* class, which is the method that reads from the file or buffer.
To solve **Challenge 1** from Section 3.4 we have implemented a producer-consumer pattern in the *ProducerConsumer* class. By doing so we have made sure that the producer can not read new data from the audio file before every connected device has been sent the currently read data from the *consume()* method. The *produce()* and *consume()* methods can be seen in Listing 1 and Listing 2, respectively.
In Listing 1 we read from an audio file and notify the consumer thread(s) that they can start and then the producer thread is told to wait. Listing 2 shows how the producer thread behaves. There exists a consumer thread for each device the master is connected to. When the produce thread has told the consumer thread it can start, each consumer thread then transmits the produced audio to each threads connected device. After each thread has transmitted it tells the producer thread it can read the next part of the audio file. This behaviour continuous until the entire

audio file has been read.

```java
public void produce() throws
     InterruptedException{
    try {
        while ((count =
             SingletonAudioStream.fileInputStream.
        read(audioBuffer)) != -1) {
            synchronized (this) {
                MainScreen.streamerList.clear();
                notifyAll();
                while
                     (MainScreen.streamerList.size()
                     <
                     ConnectionHandler.streamers)
                     {
                    wait();
                }
            }
        }
    } catch (IOException e) {
        Log.e(TAG, e.toString());
    }
}
```

Listing 1: Producer Method

```java
public void consume(OutputStream outputStream)
     throws InterruptedException {
    int id = getID();
    while(true) {
        synchronized (this) {
            while(MainScreen.streamerList.
            contains(id)) {
                notifyAll();
                wait();  }
            try {
                    outputStream.write(audioBuffer,
                         0, count);
                    MainScreen.streamerList.add(id);
            } catch (IOException e) {
                Log.e(TAG, e.toString());
            }
            notify();
        }
    }
}
```

Listing 2: Consumer Method

3.5.2.2 Slave Device

When choosing the *SlaveActivity* class the device is brought into discoverable mode, which makes it possible for other searching Bluetooth devices to locate it. Besides that it also starts the *AcceptThread* class where we open up a Bluetooth socket and wait for a device to connect. As was the case with the *MasterActivity* class as explained in Section 3.5.2.1 the *SlaveActivity* class has the same UUID as the *MasterActivity* class, else the connecting device would not get accepted.
From the *AcceptThread* class it activates the *ConnectionHandler* class which is used to read the incoming data from the connected device. Furthermore inside this class an instance of the *AudioTrack* class is called which can be used to play audio, Listing 3 shows the related code.

```java
    public static AudioTrack audioTrack = new
         AudioTrack(AudioManager.STREAM_MUSIC,
         48000, AudioFormat.CHANNEL_OUT_MONO,
         AudioFormat.ENCODING_PCM_8BIT,
         MainScreen.bufferSize,
         AudioTrack.MODE_STREAM);
```

Listing 3: AudioTrack Class

In Listing 3 the first parameter is *AudioManager.Stream_Music* and is chosen to tell the *AudioTrack* that it is music that will be played.
The second parameter, with the value of 48000, is the sample rate that it will play the received audio with. The sample rate must match that of the received audio to play the audio as intended.
AudioFormat.CHANNEL_OUT_MONO is the third parameter. It means how the device will play the audio, e.g. using one channel or more in the case of mono and stereo. This parameter also has to match the received audio file's setting of stereo or mono to play the audio as intended. Several outgoing channels can be selected for this parameter, however we will only use stereo and mono.
The fourth parameter is the bit depth, shown as *AudioFormat.ENCODING_PCM_8BIT*. In Android we only have the option to choose between 8 bit and 16 bit as the bit depth. Again the *AudioTrack's* chosen

bit depth has to match the incoming audio for it to play as intended.
MainScreen.buffersize is the fifth parameter and is the *AudioTrack's* own buffer size. The larger this is, the longer it will take to start playing the audio. When the buffer is around half filled it will begin to play the audio. In Chapter 4 we test different buffer sizes.
The last parameter *AudioTrack.MODE_STREAM* is chosen because we want to stream the music from another device, e.g. we do not have the entire audio file on the slave device it self.
The *AudioTrack* class has a write method, which we use with the received data from the connected device. By doing so the *AudioTrack* will begin to play audio when a master or master slave device has connected and the slave device's *AudioTrack* buffer becomes half filled.

3.5.2.3 Master Slave Device

As it can be observed in Figure 2 the *MasterSlaveActivity* class has the same path as the *MasterActivity* class and the *SlaveActivity* class. What happens for the master slave device is mostly just a combination of what has been explained in Section 3.5.2.1 and 3.5.2.2. Given we have these old devices, as seen in Table 5, we remove the audio output operation of the master slave device. Meaning when the master slave device is transmitting the data to a slave device the master slave device will stop playing the audio it self. This is done to avoid the older devices to bottleneck the overall performance of the system as **Challenge 3** in Section 3.4 mentions.

4 Testing

In this chapter we will go through a variety of tests conducted on the Bluetooth multi room system which we explained in Chapter 3. The point of our tests is to investigate how many devices can be connected together to form this Bluetooth multi room solution. Meaning that we will both investigate piconet and scatternet solutions at different configurations.

4.1 Testing Methodology

Recall the devices mentioned in Chapter 3.3 where we have different devices with different Android and Bluetooth versions. For every test the Huawei P10 will act as the master device, because of its more recent and powerful hardware compared to the other devices and its newer Android and Bluetooth versions.
Meanwhile the Samsung Galaxy Tab S will act as the main slave device in every test because it is the second most updated device regarding Android and Bluetooth versions. Furthermore it has the ability to play audio clearly at a bit depth of both 8 and 16 bit. Apparently some devices can not play audio clearly with a bit depth of both 8 and 16 bit but only one of them. Because of this the Samsung Galaxy Tab S will always be placed as far away from the master device as possible in a scatternet test.
With this approach we should get realistic results of how a test performs in a scatternet configuration. We only require the Samsung Galaxy Tab S to play the audio for a duration of 60 seconds without any chopping to conclude a test is successful. Given the audio is being streamed continuously and is not being stored on the device, except in a significantly smaller buffer compared to the audio file's total size, this ought to be a long enough period to notice if the audio plays with or without chopping.
As our master slave devices we decided on the Samsung GT-I9100, Samsung GT-I8190 and the Huawei U8666E. These were chosen because they all were capable of re-transmitting the received audio to one or more devices. The Samsung GT-I9100 and Samsung GT-I8190 were the most reliable when re-transmitting audio with a bit depth of 16 meanwhile the Huawei U8666E was most reliable when re-transmitting audio with a bit depth of 8. However they will be used for both bit depths when needed, since worst cast scenario is that the audio stops playing and the test has to be redone.
The remaining devices were seen unfit for the master slave role because they were unreliable with issues such as the application freezing or stopped re-transmitting for no apparent reason. These remaining devices will then be used as passive slaves, mean-

Stereo 16 Bit	Stereo 8 Bit	Mono 16 Bit	Mono 8 Bit
48000 Hz	48000 Hz	48000 Hz	48000 Hz
44100 Hz	44100 Hz	44100 Hz	44100 Hz
32000 Hz	32000 Hz	32000 Hz	32000 Hz
24000 Hz	24000 Hz	24000 Hz	24000 Hz
22050 Hz	22050 Hz	22050 Hz	22050 Hz
16000 Hz	16000 Hz	16000 Hz	16000 Hz
11025 Hz	11025 Hz	11025 Hz	11025 Hz
8000 Hz	8000 Hz	8000 Hz	8000 Hz

Table 6: Each cell represents an audio file with a specific sound configuration which are used in the tests

ing that they will be connected to a master or master slave device and receive audio, but how these devices play the audio will have no effect on our results.
Each subsection in this chapter will start by explaining how the test was conducted since it is different from each test scenario.

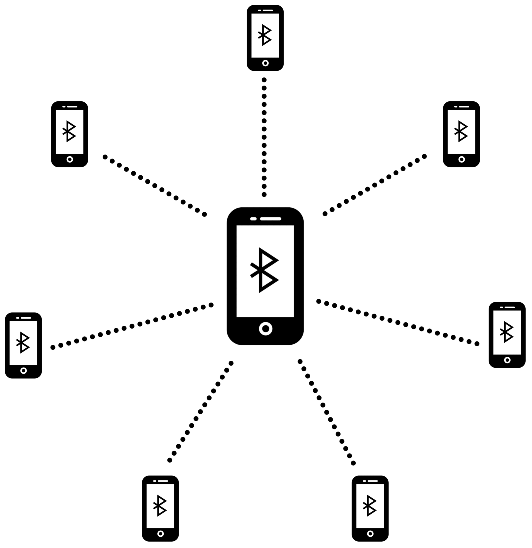

Figure 3: The centered larger phone is a master device and the seven smaller devices are slaves which together form a maximum size piconet.

4.1.1 Audio File Choice

As mentioned in Chapter 3 we use PCM to stream the audio. This means we have to chose a sound file which supports PCM. We chose the Waveform Audio File Format (WAV) and converted an audio file into several different audio files for our testing purposes. This converting was needed to test different sample rates, bit depths, stereo and mono audio files. An overview of the different audio files can be observed in Table 6. Even though we mentioned in Chapter 2 that audio with a sample rates of 11025 and 8000 Hz results in bad audio quality, we still chose to add them to the testing pool for our initial test. Besides that when we speak of sound configurations in the remainder of the chapter we refer to a specific sample rate, bit depth and if it is stereo or mono.

4.2 Piconet Test

In this section we begin by exploring the maximum number of connections with different sound configurations from a master device. Obviously the maximum number of connections we can achieve from a master device in a piconet is seven. An illustration of this test can be viewed in Figure 3. As mentioned in section 4.1 the Huawei P10 will serve as the master and the Samsung Galaxy Tab S will be our main slave. The other devices that are connected are just chosen by random, since we do not care how they play the audio.
We chose a buffer size of 8000 bytes for the initial test and use a minimum size buffer of each device for the AudioTrack. The test was conducted by first connecting the main slave to the master and keep adding slaves until the audio began to chop. The results of the piconet test can be observed in Table 7.

Next we increased the buffer to 88000 bytes to discover if a bigger buffer size can improve our initial results and solve **Challenge 2** from Chapter 3.4. Besides that we also increase the AudioTrack buffer to 88000 bytes to remove any chance of chopping sound when the connection just has been established. We only tested from a sample rate of 22050 Hz and higher this time, since going lower than this will diminish the audio quality. The results of this test can be viewed in Table 8. We also conducted tests with larger buffers

Sample Rate in Hz Bit Depth Stereo	48000 16 Bit	44100 16 Bit	32000 16 Bit	24000 16 Bit	22050 16 Bit	16000 16 Bit	11025 16 Bit	8000 16 Bit
Successful Connection(s)	0	0	0	0	0	1	2	3

Sample Rate in Hz Bit Depth Stereo	48000 8 Bit	44100 8 Bit	32000 8 Bit	24000 8 Bit	22050 8 Bit	16000 8 Bit	11025 8 Bit	8000 8 Bit
Successful Connection(s)	0	1	1	2	3	4	5	5

Sample Rate in Hz Bit Depth Mono	48000 16 Bit	44100 16 Bit	32000 16 Bit	24000 16 Bit	22050 16 Bit	16000 16 Bit	11025 16 Bit	8000 16 Bit
Successful Connection(s)	0	0	0	2	3	4	4	-

Sample Rate in Hz Bit Depth Mono	48000 8 Bit	44100 8 Bit	32000 8 Bit	24000 8 Bit	22050 8 Bit	16000 8 Bit	11025 8 Bit	8000 8 Bit
Successful Connection(s)	2	2	4	5	6	7	-	-

Table 7: Different sound configurations used to discover the maximum size of a piconet for each sound configuration. A successful connection means how many devices could be connected while keeping the Samsung Galaxy Tab S playing without chopping. The buffer size used was 8000 bytes and the buffer size of the AudioTrack was each devices' minimum buffer size.

than 88000 bytes, all up to 512000 bytes, however they seemed to severely degrade the number of successful connections.

4.3 Scatternet Test

In this section we will test various configurations of scatternets. First by testing how many master slave devices can be connected together in a chain for each sound configuration.
Secondly we will have different scenarios of scatternets where slaves will be connected either to a master device or master slave devices depending on the scenario. Given the slightly better results with a larger buffer from our piconet tests, we will continue using a larger buffer for the remaining tests. We will also continue to only test sound configurations from a sample rate of 22050 Hz and upwards.

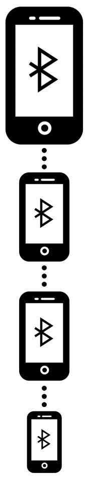

Figure 4: The largest phone is the master, the two medium size phones are master slaves, the smallest device is a slave and together they form a scatternet consisting of three piconets.

Sample Rate in Hz Bit Depth Stereo	48000 16 Bit	44100 16 Bit	32000 16 Bit	24000 16 Bit	22050 16 Bit
Successful Connection(s)	0	0	1	2	2
Sample Rate in Hz Bit Depth Stereo	48000 8 Bit	44100 8 Bit	32000 8 Bit	24000 8 Bit	22050 8 Bit
Successful Connection(s)	1	1	2	3	3
Sample Rate in Hz Bit Depth Mono	48000 16 Bit	44100 16 Bit	32000 16 Bit	24000 16 Bit	22050 16 Bit
Successful Connection(s)	1	1	2	3	3
Sample Rate in Hz Bit Depth Mono	48000 8 Bit	44100 8 Bit	32000 8 Bit	24000 8 Bit	22050 8 Bit
Successful Connection(s)	3	3	4	5	5

Table 8: Second test of maximum number of devices in a piconet. Here we focus on sound configurations at 22050 Hz and above. We also increased the buffer size and AudioTrack buffer size to 88000.

4.3.1 Master Slave Test

For this first test of a scatternet we aim to connect as many master slave devices together in a chain from the master device and connect a slave device to the last master slave. Observing Figure 4 illustrates how this test was conducted and shows a case where we have two successful master slave connections if the slave plays the audio without chopping.
The results of this test can be observed in Table 9. One can note that at every sound configuration where it has three successful master slave connections it might have been possible to increase that number. However we are limited by only having three devices that can act as a master slave device.

4.3.2 Master slave and Slave test

In this section we will test both slaves and master slaves together. Meaning that by looking at our earlier results from Table 8 and Table 9 we will suggest different scenarios for each valid sound configuration, since there is no reason to test the sound configurations where we had no successful connections and no successful master slave connections. Given we know the number of successful master slave connections and we know the maximum number of clients that can connect to a single master for each sound configuration. This knowledge was used to arrange fitting scenarios for each sound configuration. In each scenario the square represents a master device, a diamond a master slave device and a circle a slave device.
The overall purpose of these tests were to discover if it is possible to increase the number of audio playing devices in a scatternet compared to a single piconet for every valid sound configurations.

4.3.2.1 8 Bit and Stereo Configuration

First we tested the sound configuration with a bit depth of 8 and stereo. By looking at Table 9 we can note it only made sense to test from 32000 Hz and downwards, since we were not able to establish any master slave connections above 32000 Hz. The test scenarios can be observed in Figure 5. The reasoning behind these three scenarios and several other sce-

Sample Rate in Hz Bit Depth Stereo	48000 16 Bit	44100 16 Bit	32000 16 Bit	24000 16 Bit	22050 16 Bit
Successful MasterSlave Connection(s)	0	0	0	0	0
Sample Rate in Hz Bit Depth Stereo	**48000 8 Bit**	**44100 8 Bit**	**32000 8 Bit**	**24000 8 Bit**	**22050 8 Bit**
Successful MasterSlave Connection(s)	0	0	2	3	3
Sample Rate in Hz Bit Depth Mono	**48000 16 Bit**	**44100 16 Bit**	**32000 16 Bit**	**24000 16 Bit**	**22050 16 Bit**
Successful MasterSlave Connection(s)	0	0	1	3	3
Sample Rate in Hz Bit Depth Mono	**48000 8 Bit**	**44100 8 Bit**	**32000 8 Bit**	**24000 8 Bit**	**22050 8 Bit**
Successful MasterSlave Connection(s)	3	3	3	3	3

Table 9: Number of successful master slave connections at each sound configuration

narios to come is because of a sound configuration's successful connections from earlier tests. Observing Table 8 shows us the maximum number of successful slaves that can be connected to a master. Furthermore observing Table 9 we can note how many possible master slave connections are possible for each sound configuration. Given this information we can more precisely create scenarios that are worth testing.
In the case of a sound configuration with a sample rate of 32000 Hz, 8 bit and stereo we have a maximum of two successful connections to one master and two successful master slave connections at maximum. Meaning we know what is already possible and we try to increase the number of devices in the scatternet by creating different placement of extra devices. Testing scenario A, B and C from Figure 5 all ended in unsuccessful connections.

Next is the sound configuration of 24000 Hz, 8 Bit and stereo. At this configuration we can have a maximum of three slaves connected to a master and at least three possible master slave connections. The tested scenarios can be viewed in Figure 6. Scenario A, C and D all had successful connections meanwhile B and E did not.

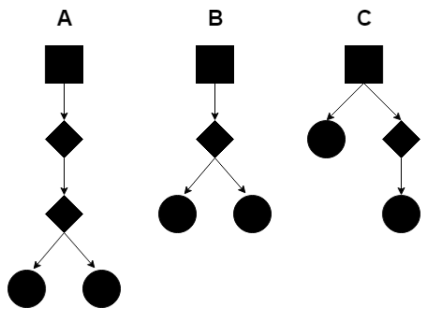

Figure 5: Three different test scenarios for a sound configuration of 32000 Hz, bit depth of 8 and stereo sound

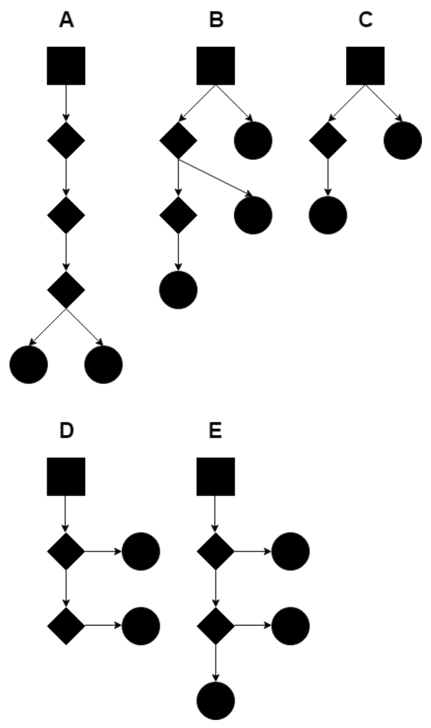

Figure 6: Five different test scenarios for a sound configuration of 24000 Hz, bit depth of 8 and stereo sound

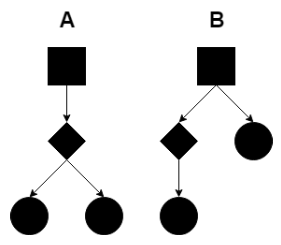

Figure 7: Two different test scenarios for a sound configuration of 32000 Hz, bit depth of 16 and mono sound

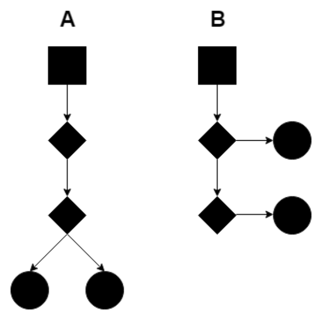

Figure 8: Two different test scenarios for a sound configuration of 24000 Hz, bit depth of 16 and mono sound

The sound configuration of 22050 Hz, 8 bit and stereo yielded in same results as 24000 Hz, 8 bit and stereo except for scenario A where it could successfully connect three slaves instead of two.

4.3.2.2 16 Bit and Mono Configuration

Now we will explain the test of a sound configuration with a bit depth of 16 and mono. We begin at 32000 Hz sample rate because this is the highest sample rate with successful master slave connections at this sound configuration. At 32000 Hz we can have at maximum of one master slave connection and a maximum of two slaves in a piconet. From this observation we test two different scenarios which can be observed in Figure 7. Both scenarios ended in unsuccessful connections.

For a sound configuration with a sample rate of 24000 Hz 16 bit and mono we can have at least three master slave connections and at maximum three slaves connected to one master. First we tested the two scenarios from Figure 7 where both scenarios ended with successful connections. Next we tested the scenarios from Figure 8 where both scenarios yielded unsuccessful connections.

Then we tested the sound configuration with a sample rate of 22050 Hz, 16 bit and mono where we have the same amount of maximum slaves and master slaves as with 24000 Hz. We tested the scenarios from Figure 8 where scenario A was successful meanwhile scenario B was not. Given it was not possible to run scenario B successfully there was no reason to develop more scenarios for testing this sound configuration.

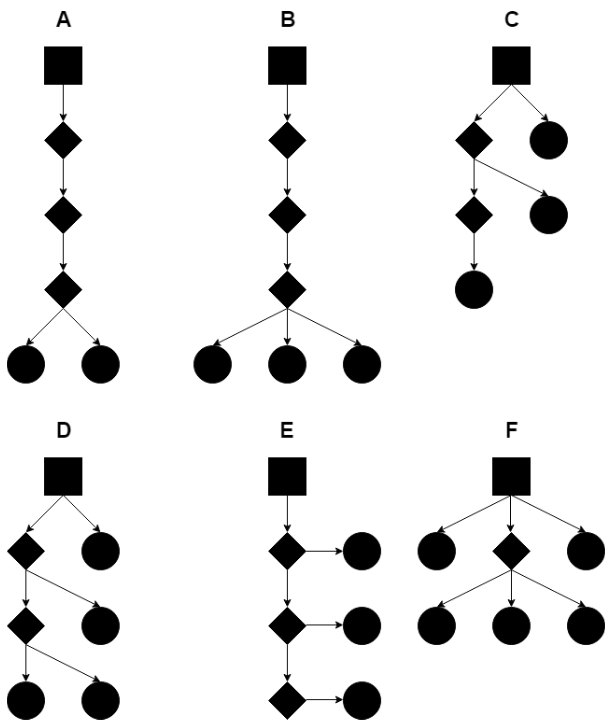

Figure 9: Six different test scenarios for a sound configuration of 48000 Hz, bit depth of 8 and mono sound

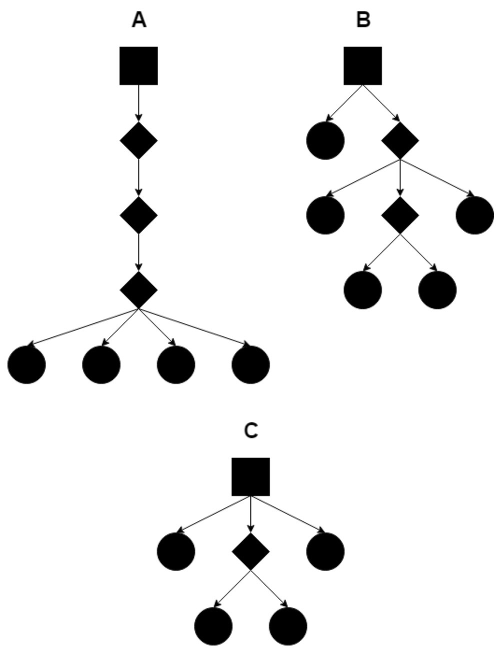

Figure 10: Four different test scenarios for a sound configuration of 24000 Hz, bit depth of 8 and mono sound

4.3.2.3 8 Bit and Mono Configuration

Lastly we tested the sound configuration of a bit depth of 8 and mono. Under this configuration we can see that it is possible to establish successful master slave connections under every tested sample rate. Therefore we began with the sound configuration with a sample rate of 48000 Hz. We know we can have a maximum of three slaves connected to a master in a piconet and we can have at least three successful master slave connections.
Given this knowledge we tested the sound configuration in six different scenarios. The different scenarios can be observed in Figure 9. Scenario A and C ended with successful connections meanwhile scenario B, D, E and F ended in unsuccessful connections.

Testing the next sound configuration, lowering the sample rate to 44100 Hz, ended with the same amount of successful and unsuccessful connections on the scenarios from Figure 9 as with a sample rate of 48000 Hz.
Next we tested the sound configuration with a sample rate of 32000 Hz, 8 bit and mono. We can have a maximum of four slaves to one master in a piconet and we can have at least three master slave connections. We tested scenario B, D, E and F from Figure 9 to discover if it is possible to establish successful connections at this sound configuration. Scenario B and E ended in successful connections meanwhile scenario D and F both yielded unsuccessful connections. The sound configuration with a sample rate of 24000 Hz, 8 bit and mono was the next test. At this sound configuration we can have a maximum of five slaves connected to one master in a piconet and at least three master slave connections. The scenarios developed for this sound configuration can be viewed in Figure 10.

Running this sound configuration on these scenarios ended with successful connections on scenario A,

B and C. We also ran scenario D from Figure 9, however that ended in an unsuccessful connection.
Applying the sound configuration with a sample rate of 22050 Hz, 8 bit and mono on scenario D from Figure 9 ended in a successful connection. Given we do not have more devices available for testing we can not develop more meaningful scenarios for this sound configuration.

4.3.2.4 Summary

To get an overview of what the different scenarios taught us we present Table 10. Keep in mind that only seven devices were used for this test. Meaning that we do not know for the sound configurations with a sample rate of 24000 Hz and 22050 Hz, 8 bit and mono could have a higher number of maximum connections in a scatternet.

5 Discussion

In this chapter we will discuss the results we have obtained through several test scenarios from Chapter 4. Furthermore we will use the information gathered about Bluetooth and audio from Chapter 2 and analyze our results according to the theory.

5.1 Discussion of Piconet Test

In this section we will discuss our results from our two piconet tests from Chapter 4.2 and compare them together to understand possible sizes of piconets and that different size buffers give different results.

5.1.1 First Piconet Test

Figure 11 illustrates the results we got from the first piconet test in Chapter 4.2 from Table 7. This shows us the number of successful connections a master device can have with a buffer size of 8000 for every sound configuration we have tested.
We can observe from Figure 11 that to achieve a full size piconet, seven slaves and one master, is only possible with a sample rate of 16000 Hz and lower, a bit depth of 8 and mono. However according to Chapter 2.1 playing audio at these sample rates result in very poor audio quality. If we then consider an old popular sound file, the MP3, which has a sample rate of 22050 Hz we can connect six slave devices to our master device, but still with a bit depth of 8 and mono.
At a sound configuration with a sample rate of 44100 Hz, CD sample rate, 8 bit and mono it is only possible to connect two slave devices. However lowering the sample rate to 32000 Hz, which according to Chapter 2.1 is a good compromise to make to only lose a little audio quality makes it possible to connect four master slave devices to a single master device.
For a sound configuration with 8 bit and stereo sound we can observe in Figure 11 that the number of slave devices that can be connected to a master device is significant lowered at each sample rate. The maximum size of slave devices at a poor audio quality is five devices. At MP3 quality we can connect up to three slave devices and we can only connect one slave device at both 32000 Hz and 44100 Hz.
Therefore by using stereo sound, in a sound configuration with a bit depth of 8 and a sample rate between 8000-48000 Hz, causes the amount of connections to slave devices from a master device to be lowered considerably by up to 75 % compared to using mono sound.
Considering the sound configuration with a bit depth of 16 and mono from Figure 11 shows that it is somewhat similar to the sound configuration with a bit depth of 8 and stereo. At the poor audio quality sample rates we can connect five or four slave devices to a master device, meanwhile for MP3 quality audio we have identical amount of slave devices as with the sound configuration with bit depth of 8 and stereo. However the higher quality audio can not have any slave devices connected to a master device, e.g. from a sample rate of 32000 Hz and upwards.
This is the first sound configuration with a bit depth of 16 which, at least for this test, shows that streaming an audio file with a bit depth of 16 and mono sound is similar to streaming an audio file with a bit depth of 8 and stereo at certain sample rates. Besides that by increasing the bit depth from 8 to 16 in a mono configuration severely lowers the amount of slaves that can connect to a master device and removes the possibility for streaming at a higher sample rate to a slave device.

Sample Rate in Hz Bit Depth Stereo	48000 8 Bit	44100 8 Bit	32000 8 Bit	24000 8 Bit	22050 8 Bit
Maximum Connections in Scatternet	-	-	3	5	6
Sample Rate in Hz Bit Depth Mono	**48000 16 Bit**	**44100 16 Bit**	**32000 16 Bit**	**24000 16 Bit**	**22050 16 Bit**
Maximum Connections in Scatternet	-	-	2	4	4
Sample Rate in Hz Bit Depth Mono	**48000 8 Bit**	**44100 8 Bit**	**32000 8 Bit**	**24000 8 Bit**	**22050 8 Bit**
Maximum Connections in Scatternet	5	5	6	7	7

Table 10: Maximum number of connections in a scatternet for each sound configuration

Lastly we have the sound configuration with a bit depth of 16 and stereo sound. Observing Figure 11 shows us the difficulties of this sound configuration. One can quickly note that it's maximum number of connections to a slave is three, two and one at a poor audio quality. Furthermore it does not support audio streaming to a slave device at MP3 quality audio or better. Comparing this to our other sound configurations tells us that streaming an audio file with 16 bit and stereo lowers the amount of connections to slave devices massively and only audio at a poor quality is possible to stream to one or more slave devices.

Given these results we know how many slave devices we can connect to a master slave device with any of the sound configurations. With this knowledge and the bytes per second needed to sample audio from Table 3 in Chapter 2 we create the following graph which can be observed in Figure 12. This graph shows the amount of bytes per second needed to sample audio for each successful sound configuration in the constructed piconet. Meaning the master device has to feed the piconet with a certain amount of bytes per second for it to stream continuously. At a sample rate of 22050 Hz and at each combination of bit depth and stereo/mono, except the 16 bit and stereo sound configuration, the connected slave devices require together 132300 bytes per second to play the audio file successfully.

Raising the sample rate significantly lowers the bytes per second and amount of devices for the 8 bit stereo and 16 bit mono sound configurations. However the 8 bit mono sound configuration still remains at a fairly high bytes per second until it drops a bit down at a sample rate of 44100 Hz. Observing the lower end of the sample rate, for example at 8000 Hz, we have the 8 bit mono sound configuration only demanding 56000 bytes per second. This tells us if Bluetooth supported more than eight devices in a piconet we could most likely connect more slave devices at this configuration since it is only half of the bytes needed at a sample rate of 16000 Hz.

Another interesting occurrence is if we look at the sound configurations of 8 bit stereo and 16 bit stereo both needing 64000 bytes per second at each their sample rates of 32000 Hz and 16000 Hz respectively and we look at the bytes per second, 128000, supported by 16 bit mono and 8 bit mono at a sample rate of 16000 Hz and 32000 Hz respectively. One should think why these two sound configurations can not establish double the amount of slaves they have at each sample rate to match the other two sound configurations. This behaviour is most likely caused by the max payload size of 1021 bytes in a classic Bluetooth connection. Meaning that it is just not possible to establish a high enough throughput for the audio files with higher bit depth and stereo for several connections. This claim is backed up by also not being able to establish one single connection with

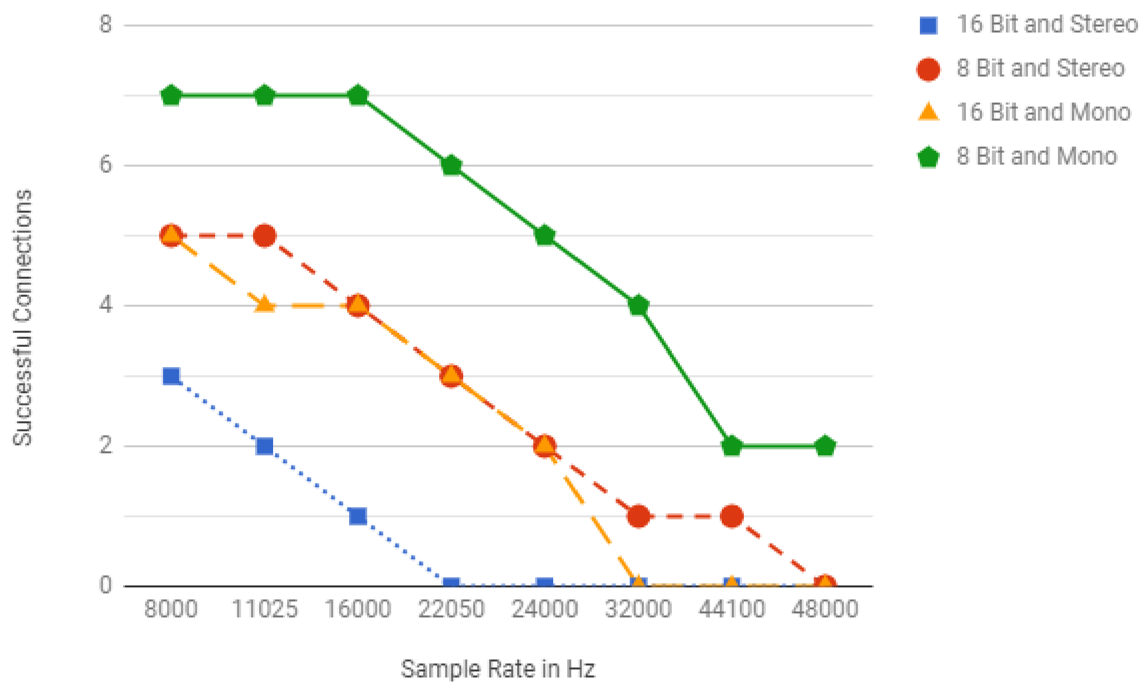

Figure 11: Graph showing the amount of successful connections at each sample rate from the first piconet test.

a sound configuration of 32000 Hz, 24000 Hz, 22050 Hz, 16 bit and stereo. These configuration needs respectively 128000, 96000, 88200 bytes per second to sample audio, which should be possible according to our maximum achieved bytes per second, 132300, in a piconet. However as the results show this was not possible to achieve.

5.1.2 Second Piconet Test

For our second test of piconets we raised the buffer from 8000 bytes to 88000 bytes and the AudioTrack's buffer size to the same amount of bytes. Besides that we also only care for an audio quality of MP3 or higher this time. Our results are presented as a graph in Figure 13. One can quickly note that there exists a better coherence between the sound configurations now, especially between 8 bit stereo and 16 bit mono. Considering 16 bit and stereo it is now possible to connect two slave devices at 22050 Hz and 24000 Hz, which makes it possible to stream MP3 quality audio. Besides that one slave device can be connected at a sample rate of 32000 Hz. Therefore by increasing the buffer to 88000 bytes from 8000 bytes it makes it possible to stream audio at a higher audio quality at 16 bit and stereo.
At 8 bit and stereo we get an improvement at 24000 Hz where it is now possible to connect three slave devices. 32000 Hz now supports two slave devices and at 48000 Hz one slave device is now viable.
16 bit and mono now supports three slave devices at 24000 Hz and two slave devices at 32000 Hz. Furthermore it is also possible to connect one slave device at 44100 Hz and 48000 Hz. This sound configuration also matches the 8 bit stereo sound configuration now with regards to possible slave devices. This was ex-

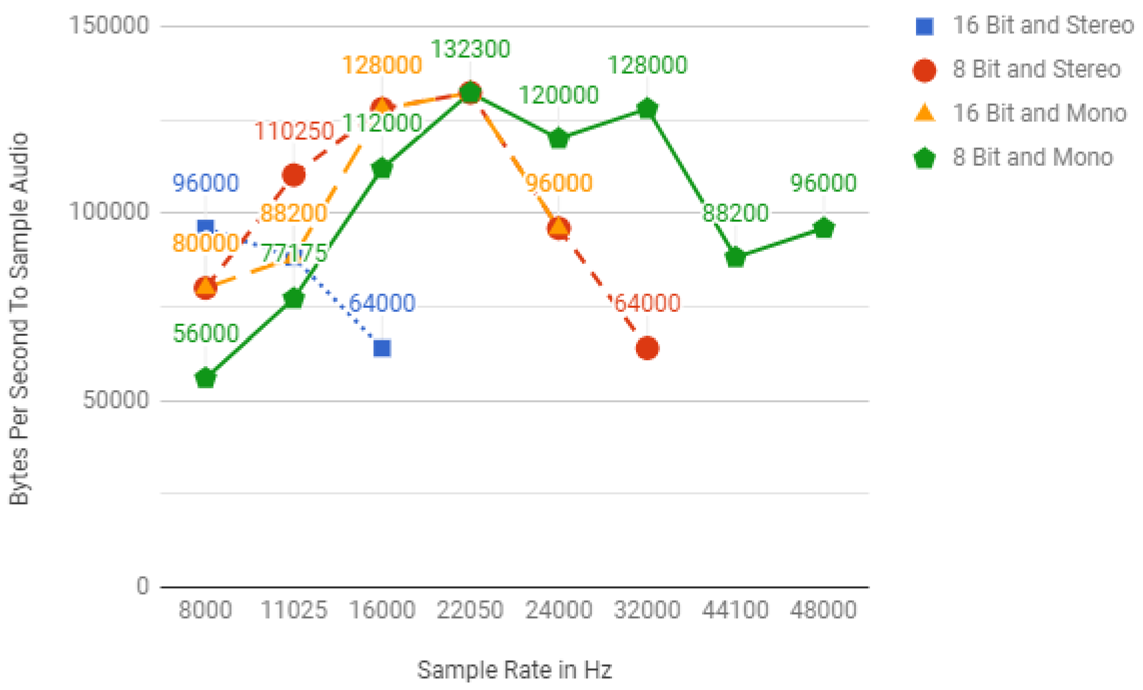

Figure 12: Graph showing the amount of bytes the master device can support a piconet with at each sound configuration from the first piconet test.

pected since they require the same amount of bytes per second at every sample rate. However we did not get that result from the first test. So clearly it is more suitable for the sound configuration of 16 bit and mono to have a buffer size of 88000 bytes compared to 8000 bytes.
At last we have 8 bit and mono where at 22050 Hz we get one slave device less compared to the first test. However at 44100 Hz and 48000 Hz the amount of slave device is raised to three. Therefore at this sound configuration we lose one slave device at MP3 quality, but gain one slave device at CD sample rate.
Figure 14 shows us a graph where we can note that the bytes per second has raised significantly compared to Figure 12 where a smaller buffer size was used. This means by raising the buffer to a certain size we can, in most of the cases, connect more slave devices to a master device. At 24000 Hz, 16 bit and stereo we had two slave devices connected to a master, which requires 192000 bytes per second to sample. This number of bytes is significantly larger than the 132300 bytes we saw from our first test which was the maximum amount of bytes. Furthermore the three other sound configurations at 44100 Hz, 48000 Hz and to some degree 32000 Hz are reaching a considerable higher bytes per second compared to the initial test. This means by using a bigger buffer it is possible to increase the number of slave devices to a master for most of the sound configurations. Therefore The master phone can supply a higher amount of bytes per second to a piconet when using a buffer size of 88000 bytes compared to 8000 bytes.
Al though it is possible to feed a piconet with 192000 bytes per second it is still not possible to establish one slave device to a sound configuration of 44100 Hz and 48000 Hz at a bit rate of 16 and stereo sound. At 44100 Hz it needs 176400 bytes per second to sample audio, but still this is unsuccessful. We might still

be limited by the maximum payload size of 1021 or that the data just can not be sent fast enough to a slave device for it to sample it for the highest audio quality.

5.1.3 Summary

From these first two piconets tests we have shown that a buffer size of 88000 bytes for streaming audio over a classic Bluetooth connection is the better option compared to a buffer size of 8000 bytes. In most cases extra slave devices were gained by using the larger buffer except for one case with the sound configuration 22050 Hz, 8 bit and mono where one slave device was lost.

5.2 Discussion of Master Slave Test

In this section we will discuss the findings from the master slave test from Chapter 4.3.1 which are illustrated in Table 9. The purpose of this test was to discover if it is possible to increase the amount of slave devices that receive audio from a master device by creating a scatternet. Furthermore when we speak of a scatternet in this section we mean in such a configuration as illustrated in Figure 4 from Chapter 4
We begin by looking at the sound configuration with a bit depth of 16 and stereo sound. Here we can see that no master slave connection is possible which is surprising since we know from Figure 13 that two slave devices were possible at 22050 Hz and 24000 Hz. The reason for this could be that the master slave devices that were used are bottle necking and causing a severe delay when re-transmitting the data compared to the master device. Therefore by creating a scatternet to increase the amount of devices only lowers the number of devices at this sound configuration.
Next we look at the sound configuration with a bit depth of 8 and mono sound. Here we can observe that creating a scatternet for 44100 Hz and 48000 Hz does not work. However by doing so for 32000 Hz, 24000 Hz and 22050 Hz increases the amount of devices by one. Furthermore one should also note we only tested this with three master slave devices, meaning that it might be possible to increase the amount of devices for 22050 Hz and 24000 Hz even further. Therefore creating a scatternet will increase the amount of connected devices for this sound configuration at the sample rate of 22050 Hz, 24000 Hz and 32000 Hz.
Now we look at the sound configuration with a bit depth of 16 and mono where we expect similar results compared to 8 bit and stereo because of the identical amount of connections which can be observed in Figure 13. The results are as expected the same, except for 32000 Hz where it is only possible to establish one master slave connection. Why this occurs might be because of a master slave device's performance. However the same master slave combination was used for the 8 bit mono sound configuration. None the less it still creates at least an extra device for 22050 Hz and 24000 Hz by creating a scatternet, which is an improvement compared to a piconet.
For the last sound configuration with a bit depth of 8 and mono sound we also see improvement. For the sample rates of 48000 Hz and 44100 Hz it is possible to connect all three master slave devices, which gives an extra device compared to the piconet. Again we are limited by the amount of master slave devices to test if more devices could be connected. Regarding 32000 Hz, 24000 Hz and 22050 Hz we can not improve the number of devices by use of a scatternet since the number of connections in a piconet is equal or surpasses the number of devices we can use in this scatternet setting. However by creating this scatternet still yielded in improvement regarding 48000 Hz and 44100 Hz compared to a piconet.
To illustrate the results more clearly we can observe Figure 15, which shows us the bytes per second a scatternet can supply, and compare it to Figure 14, which shows us the bytes per second a piconet can supply. Observing 8 bit and stereo sound in both figures shows that the byte supply is significant higher between 22050 Hz and 32000 Hz. Meanwhile the same is true at 22050 Hz and 24000 Hz. Keep in mind that at 24000 Hz and 22050 Hz we were limited to test further. The same can be said of the sound configuration of 8 bit and mono sound. Here the bytes per second increases significantly at 44100 Hz and 48000 Hz, meanwhile when lowering the sample rate it actually drops lower than the piconet bytes per second. However this is due to not having more master slave

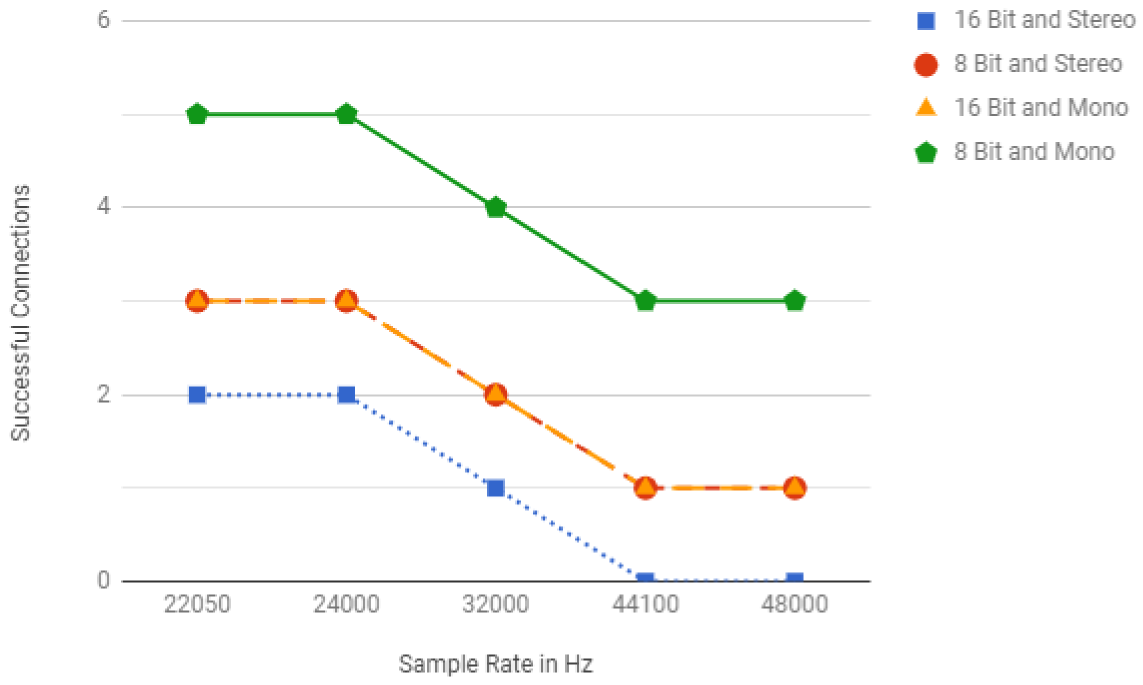

Figure 13: Graph showing the amount of successful connections at each sample rate from the second piconet test.

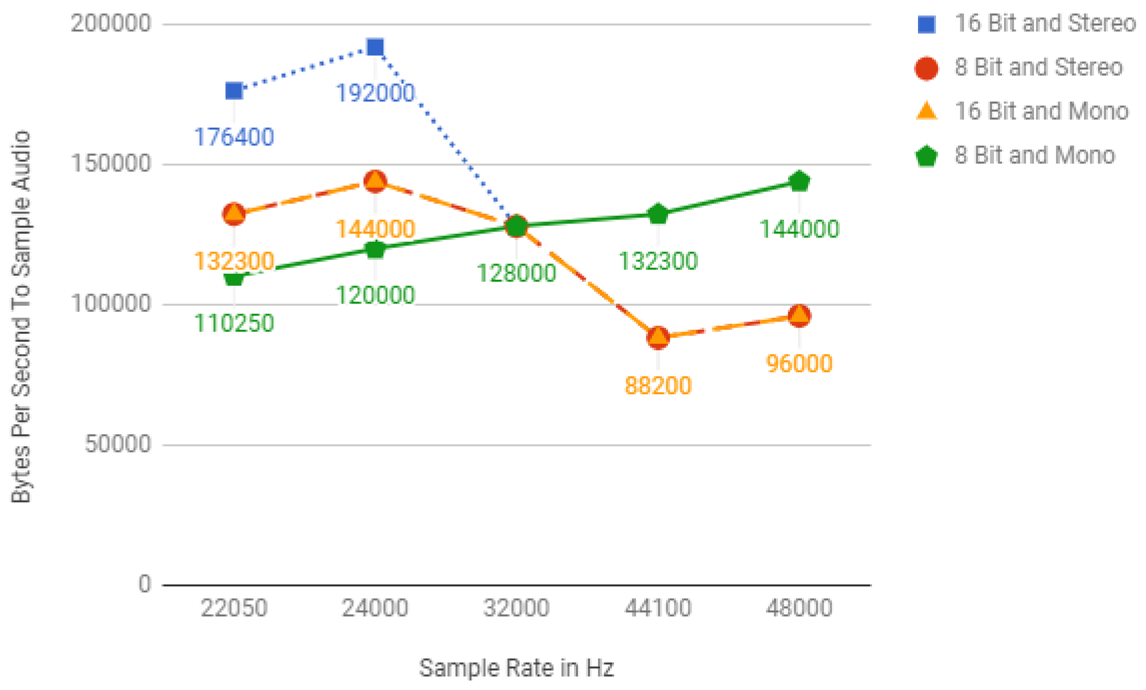

Figure 14: Graph showing the amount of bytes the master device can support a piconet with at each sound configuration from the second piconet test.

devices to connect.

5.2.1 Summary

From this test we have confirmed that it is possible to increase the amount of connected devices by creating a scatternet, e.g. a scatternet that can supply higher bytes per second than a piconet to feed more connected devices. Al though not for the following sound configurations: 24000 Hz and 22050 Hz with a bit depth of 16 and stereo sound and 32000 Hz with a bit depth of 16 and mono sound.

5.3 Discussion of Master Slave and Slave Test

In this section we will discuss our findings from Chapter 4.3.2 where we had different scatternets for each sound configuration. This test was done to identify if it is possible to improve on the results from Chapter 4.3.1 and Table 9 by combining master slave and slave devices in different setups.

5.3.1 8 Bit Stereo Test

We begin at a sample rate of 32000 Hz. Scenario A in Figure 5 with four connected devices was tested and failed meanwhile scenario B and C from the same figure also failed, even though only three devices were connected in the last two scenarios. This means we can not connect more than three devices with this sound configuration and how the scatternet is constructed also has an effect on how many devices can be connected.
At a sample rate of 24000 Hz was tested with scenario A,C and D in Figure 6 which were successful

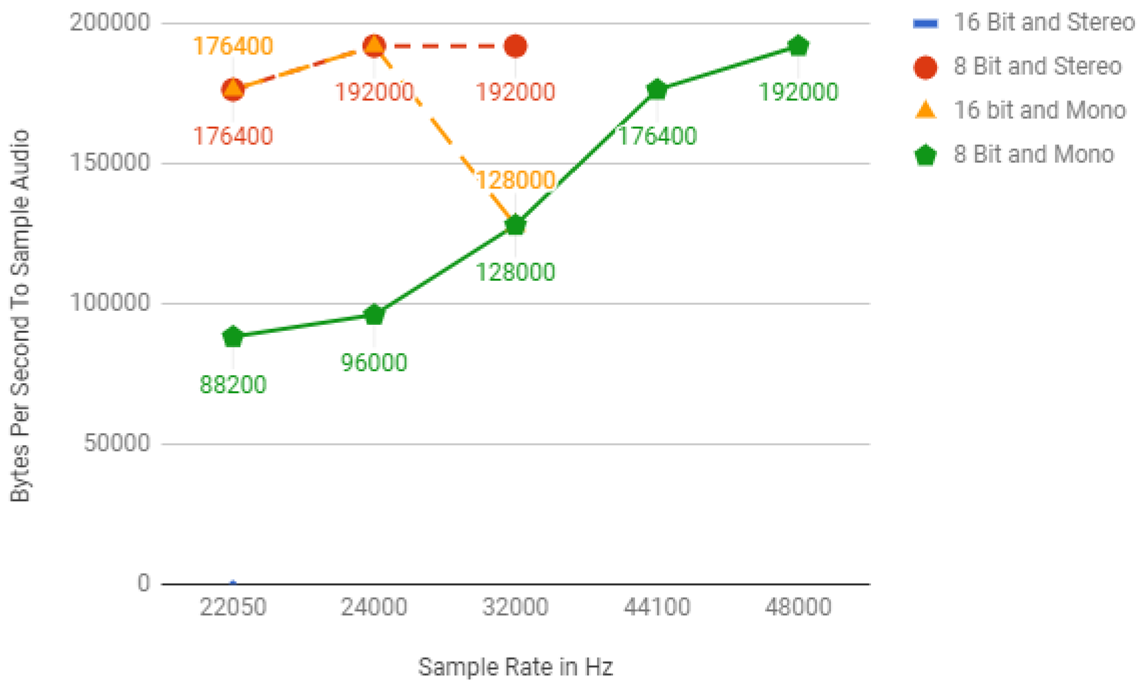

Figure 15: Graph showing the amount of bytes that can be supported from the initial master slave test at each sound configuration.

while scenario B and E failed. Meaning at this sound configuration we can connect up to five devices and how the scatternet is constructed still has an impact of the amount of connected devices.
Applying a sample rate of 22050 Hz to the same scenarios as 24000 Hz gave the same results and was able to connect one more slave device to scenario A. This gives the 22050 Hz sound configuration the possibility to connect up to six devices in a scatternet. As before it is possible to increase the amount of connected devices by constructing a scatternet in a certain way.

5.3.2 16 Bit Mono Test

Regarding the sample rate of 32000 Hz we tested both scenario A and B in Figure 7 which were unsuccessful, meaning that it is not possible to improve the amount of connected devices by using a scatternet for this sound configuration. This is a surprising result given at a sound configuration of 32000 Hz, 8 bit and stereo has a maximum of connected devices of three and at this sound configuration only two is possible. Given both of them require the same amount of bytes per second this difference might be because of the master slave device's performance. Therefore by creating a scatternet for this sound configuration does not increase the amount of connected devices.
At a sample rate of 24000 Hz we tested scenario A and B from Figure 7 where both were successful while scenario A and B from Figure 8 were unsuccessful. This shows that we can not establish more connections than we already know was possible from the master slave test. Meaning by creating different scatternets will not improve the amount of connections for this sound configuration.
Lastly is the 22050 Hz sample rate configuration.

This sound configuration succeeded at scenario A but failed at scenario B from Figure 8. Which means we can not increase the amount of connections by constructing different scatternet scenarios. Given these last two sound configurations' results we suspect the master slave device is the reason for not being able to increase the amount of connected devices.

5.3.3 8 Bit Mono Test

With a sample rate of 48000 Hz scenario A and C from Figure 9 was successful meanwhile Scenario B, D, E and F was not. Resulting with the amount of connections have improved to five connections through these scenarios, which means that is the maximum amount of connectable devices for this sound configuration. At a sample rate of 44100 Hz we saw the same results as with a sample rate of 48000 Hz.
The sound configuration with the sample rate of 32000 Hz we tested the scenarios that were unsuccessful for 44100 Hz to 48000 Hz from Figure 9 which resulted with scenario B and E was possible meanwhile scenario D and F was not. This means we can increase the amount of connected slave devices to six by constructing certain scatternets. At a sample rate of 24000 Hz we ran scenario A, B and C from Figure 10 all were successful meanwhile scenario D from Figure 9 was unsuccessful. Meaning that we can increase the amount of devices by applying certain scatternets at this sound configuration and increased the device count to seven. However it might be possible to connect more than seven devices, but we do not have more devices to test with. Applying the sound configuration of 22050 Hz on scenario D from Figure 9 gave us a successful connection.
By observing Figure 16 and comparing it to Figure 15 we can see that it is possible to increase the amount of bytes per second a master device can supply a scatternet with, e.g. connecting more devices. However at certain sound configurations such as 16 bit stereo and 32000 Hz, 8 bit and mono, it was not possible to increase the amount of connected devices by testing different scenarios. 264600 bytes per second is the maximum amount of bytes a master device can supply to a scatternet from our test. Al though we can not be certain if this is the real maximum since we ran out of devices for finding the maximum at the sound configuration of 22050 Hz and 24000 Hz, 8 bit and mono.

5.3.4 Summary

To summarize this section it is possible to increase the amount of connected devices in a scatternet at certain sound configurations. Besides that how the scatternet is constructed also has an impact of the amount of devices that can be connected.

5.4 Final Summary

Given the discussion of all of our results so far has shown us for several sound configurations the maximum amount of slave devices in a piconet and how this is possible to increase by creating several piconets forming a scatternet.

6 Conclusion

In [5] we investigated the Bluetooth technology and conducted several tests regarding the range of Bluetooth when audio streaming. From the obtained knowledge we proposed the idea of a Bluetooth audio streaming multi-room system.
In this paper we then implemented a Bluetooth audio streaming multi-room solution and tested it with regards to audio quality and number of connectable devices.
Our results showed that it is not possible to stream audio at the highest CD quality, 44100 Hz, 16 bit and stereo, to a single device over Bluetooth. However by lowering the audio quality with either mono or lowering the bit depth to 8 makes it possible to stream audio to one device. Furthermore to stream audio to three devices from a single device can be done by using both mono and a bit depth of 8. We were able to improve the number of devices to five when streaming audio with a bit depth of 8 and mono with use of a scatternet.
Streaming audio at a 32000 Hz, 16 bit and stereo,

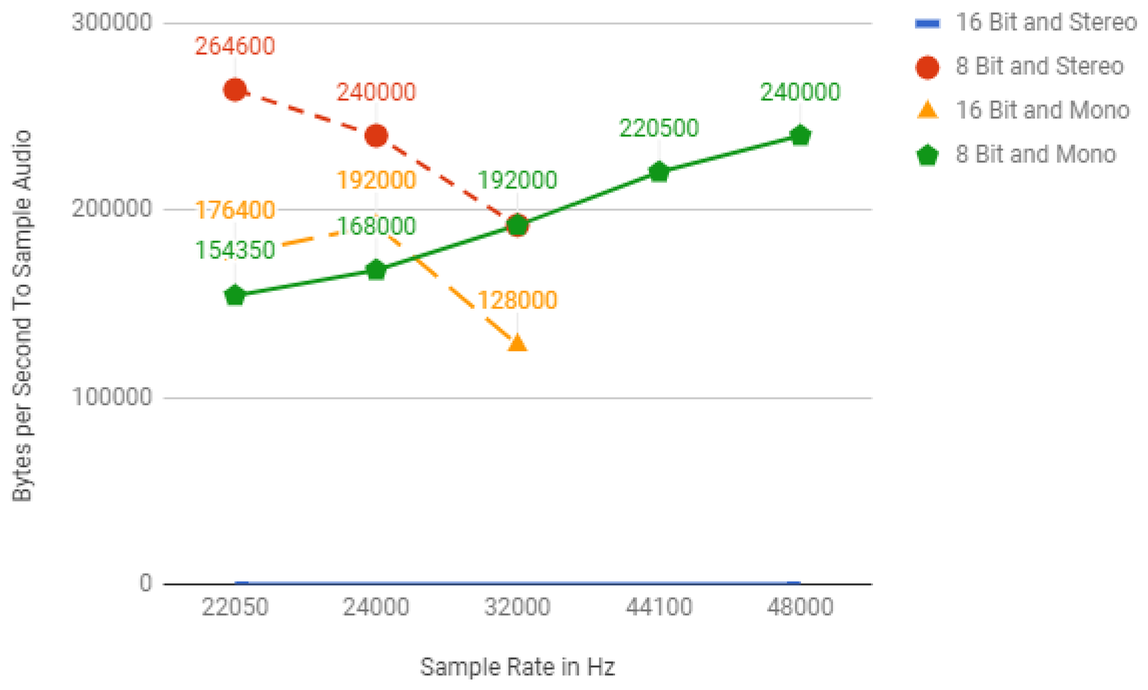

Figure 16: Graph showing the maximum amount of bytes that is supported at each sound configuration in a scatternet.

it was possible to stream to one device and lowering the bit depth to 8 made it possible to stream to two devices simultaneously, Which also was the case by changing stereo to mono. Lowering both the bit depth to 8 and stereo to mono resulted with four possible connections. Employing a scatternet here improved the number of devices to three for a bit depth of 8 and stereo and the number of devices to six for a bit depth of 8 and mono.
Streaming at MP3 quality, 22050 Hz, 16 bit and stereo, it was possible to stream to two devices and to three devices when lowering the bit depth to 8 or employing mono instead of stereo. Lowering both of these meant it was possible to stream to five devices. Applying a scatternet improved the number of devices from three to six at a bit depth of 8 and stereo meanwhile a bit depth of 16 and mono improved from three devices to four devices. Furthermore employing a scatternet for a bit depth of 8 and mono improved from five devices to at least seven devices.
Therefore we can conclude that it is possible to create an audio streaming multi-room solution over a Bluetooth connection, but at a cost. Meaning it is not possible to stream audio at the highest CD quality and having to degrade the overall audio quality to connect multiple devices.

7 Future Work

What could be intriguing to further investigate would be to test it full scale, meaning placing the devices in several rooms which would give more realistic results.
Another interesting approach would be to implement the system on a hardware device that is more suitable for transmitting and has a stronger Bluetooth

chip, e.g. power class.
Given the launch of Bluetooth 5 last year it would also be compelling to test the system with only Bluetooth 5 devices because of the supposedly higher bit rate. Another idea would be to implement the master device to broadcast the audio instead of sending it to specific MAC addresses.

References

[1] Audacity. Sample rates. `https://wiki.audacityteam.org/wiki/Sample_Rates`. Accessed: 2018-30-04.

[2] L-J Chen, Rohit Kapoor, Kevin Lee, MY Sanadidi, and Mario Gerla. Audio streaming over bluetooth: an adaptive arq timeout approach. In *Distributed Computing Systems Workshops, 2004. Proceedings. 24th International Conference on*, pages 196–201. IEEE, 2004.

[3] Roy Friedman, Alex Kogan, and Yevgeny Krivolapov. On power and throughput tradeoffs of wifi and bluetooth in smartphones. *IEEE Transactions on Mobile Computing*, 12(7):1363–1376, 2013.

[4] Google. Bluetooth overview. `https://developer.android.com/guide/topics/connectivity/bluetooth`. Accessed: 2018-30-04.

[5] Michael Jensen. A proposed multi-room system over a bluetooth network. 2018.

[6] Ben Joan. Difference between mono and stereo. `http://www.differencebetween.net/technology/difference-between-mono-and-stereo/`. Accessed: 2018-30-04.

[7] Bluetooth SIG. A2dp profile. `https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=303201`. Accessed: 2018-30-04.

[8] Wikipedia. Audio bit depth. `https://en.wikipedia.org/wiki/Audio_bit_depth`. Accessed: 2018-30-04.

[9] Wikipedia. Pulse-code modulation. `https://en.wikipedia.org/wiki/Pulse-code_modulation`. Accessed: 2018-30-04.