Networked Remote Music Collaboration Using Machine Learning

Nishtha Nayar[1,*] and Divya Lohani[2]
Department of Computer Science & Engineering
Shiv Nadar University
Gautam Buddha Nagar, U.P., India
Email: [1,*]nn469@snu.edu.in, [2]divya.lohani@snu.edu.in

Abstract—**Communication using mediums like video and audio are essential for a lot of professions. In this paper, interaction with real time audio transmission is looked upon using the tools provided by the domains of IoT and Machine Learning. The transport layer protocols – TCP and UDP are examined for audio transmission quality. Further, different Recurrent Neural Networks (RNN) models are examined for their efficiency in predicting music in case of loss of packets during transmission. These predictions fill in the gaps and help prevent any lag in a real time performance/jam where the loss of musical notes can be a hindrance.**

Keywords— Remote Music Collaboration, Machine Learning, Internet of Things, TCP, UDP, Recurrent Neural Networks

I. Introduction

As the entertainment industry is growing at a rapid rate, the need for bringing together people from across the world is also increasing to enable bigger and wider projects in the industry. This need for collaboration is majorly seen in the music industry where are musicians seeking out to create music with others who might be situated at a far-way location. One of the main problems which arises in this scenario is the geographical distance which might discourage an interested group to collaborate as it would turn out to be an expensive procedure. The same problem can even be seen with concerts where musicians are expected to travel large distances to be present at the main location to be able to perform together. [1] Research work [2] as well as software [3] to tackle this problem are already been worked upon. Music collaboration has been an interesting research topic for very long and various applications and technologies have been built for the same.

The idea of music collaboration requires audio to be transmitted over the internet over large distances and the advancement of Internet of Things can help monitor and control music equipment across these distances. IoT can be applied to attach sensors to instruments to record their performance which can further be transmitted remotely, in real time. It can further help manage the large amount of data artists produce and can keep help in remote mixing and recording.

The main motivation behind this work is to take steps towards bridging the gap between musicians/artists who are not present together physically and enable them to pursue their art and collaborate together. Hence, we examined this common problem and how to the various technologies available today in the domain of music; enabling the synergy between art and technology. It is essential to see how these technologies are impacting the music industry and the constraints on it and how these can be improved to benefit the musicians further. In this work, there are two main things which we have focused upon. The first has been examining the transport layer and comparing two protocols - TCP and UDP to find out which is the better one for sending audio with less latency and loss. This was done using java socket programming of various test cases along with using a software for catching and analyzing the packets. For the second part, the problems which occur at the end point of the communication were looked upon. One of these problems was the delay/disturbance of the music being received at the receiver side. If the receiver is not able to hear what the sender is sending in real time, it would prevent him/her from responding with the appropriate music which would affect the quality music being created as well as the experience of the musicians. Hence models were trained to predict a few bars of notes which can prove as an approximate substitute for the missing bars of notes which have not been transported properly and to keep the flow of the jam/performance going coherently.

The rest of the paper is organized as follows: Section II overviews the current state of the art solutions for this problem, Section III details the experimental design and the methodology followed, Section IV discusses the results, and Section V concludes the findings and outlines how this problem can be solved further.

II. Related Works

Daisyphone is a software which has been developed for remote music collaboration [4]. The basic approach is to loop and synchronously update music. This provides semi-synchronous collaboration with low-bandwidth requirement.

There are various approaches for transmitting audio over a network. As seen in the paper by Center for Computer Research in Music and Acoustics written by Chris Chafe [5], various experiments using WAN, LAN networks, protocols like TCP, UDP have been done to test the audio transfer quality over physical distances. As per the research, the quality of service (QoS), latency, delay are the technical difficulties which one comes across. The streaming engine is a support system for various applications and it contains classes and plugins under the four categories: input, output, processing and input mixing.

978-1-5386-8235-7/18/$31.00 ©2018 IEEE

Another paper by the same team looks at the physical model synthesis which focuses on the use of internet acoustics [6]. Geographically distributed physical models have been used to 'ping' internet connections between two network hosts. A simple plucked string model consisting of two unit generators, a delay line and a low-pass filter, forming a feedback loop which becomes a hi-Q, IIR resonator (UG is a term used in music synthesis for a signal processing element). An input is provided to excite the loop resonance. Excitation is a "pluck-like" pulse signal which is triggered either in real-time by user interaction, or from a process such as a musical score. Software synthesizers run the model in real time sending samples to computer's sound output.

Time delay has also been used to create musical devices to play with and on the network [7]. With short delay, a network is shown to be used as a reverberation medium. This technique has the advantage of making the sound distance and delay audible to the listener. Hence, the delay can be used to improve the sound and experience rather than pose as a problem. This is further used to create network meshes for distributing rhythmic patterns over a network. An important technique which can be used is Feedback-Locking. A local sound is fed-back once (or more) to the original location after it reaches the remote node.

There have been attempts to use statistical models to predict notes [8]. Prediction is used to beat the time delay and enhance the synchronization of the audio being transferred. The basic idea is 'performance-guided synthesis' in which a very convincing model takes its cues in real time from a performer.

Development of JackTrip [9], an application for music collaboration shows that it is the total delay between sound capture and sound projection which counts. This splits out into (i) acoustic (air path) delays, e.g., the distance between the instrument and the capture microphone and between the speakers and ears; (ii) analog to digital and digital to analog conversion (ADC/DAC) delay, i.e., the time it takes for an analog source to be transformed into digital and back; (iii) settings chosen for audio quality and packetization, including audio sampling rate and bit depth resolution, buffer and packet sizes, and others; (iv) network transmission delays, including physical (geographical) distance, transmission delays induced by switches, routers, firewall and network congestion.

A lot of research has also been done in ML and DL for generating music and melodies, especially LSTM. Clara [10] is an AWD-LSTM that composes piano music and chamber music. Although a lot of models insist on a small note range, or in having a limited or fixed number of notes per step, this model uses a 62-note range (which encompasses most of classical piano and violin music), and allows any number of notes and instruments at a time. It uses both chord-wise and note-wise encoding. This paper [11] also deals with LSTM and hierarchical decoding. The researchers have proposed MusicVAE, a recurrent Variational Autoencoder which utilizes a hierarchical decoder for improved modeling of sequences with long-term structure. Studies have also been done to predict polyphonic midi files using this method [12]. There have been attempts to modify this model, Douglas Eck and Jasmin Lapamle [13] trained their model with the bias of finding correlations with align with the meter of the piece.

Researchers at Stanford have studied various other ways of music composition using Naive Bayes, neural network and encoder-decoder models [14]. They found that LSTM RNN model performed the best while Naive Bayes and neural network over-fitted the data. The encoder-decoder model composed very repetitive results.

One process for which Machine Learning and Deep Learning have been used similar to generating music is chord detection. Xinquan Zhou and Alexander Lerch have worked on evaluating the results for various methods and configurations, including input pre-processing, a bottleneck architecture, and SVMs vs. HMMs for chord classification [15]. Scholars at National Taiwan University have worked on something similar while extending the methods to real-time retrieval and recommendation of similar songs [16]. Recurrent neural networks have been focused upon and there has been previous done for this as well. In a paper by Google [17], they've proposed the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The two different approaches word-RNN and char-RNN have also been researched at IRCAM [18] and they have created a word-RNN model to generate jazz chords and melodies. Basic RNN has been modified to biaxial RNN [19] where the idea is of having two axes (and one pseudo axis): there is time axis and the note axis.

Research was also done to add convolution to neural networks [20] which also showed that sample quality improves significantly when we use blocked Gibbs sampling to iteratively rewrite parts of the score.

III. Experimental Design

A. Networking Using TCP and UDP

Transmission Control Protocol (TCP) is a transport protocol which enables two hosts to establish a connection and exchange streams of data. TCP guarantees delivery of data and also guarantees that packets will be delivered in the same order in which they were sent. Whereas User Datagram Protocol (UDP) uses connectional communication and transfers the data using datagrams. UDP is suitable for purposes where error checking and correction are either not necessary or are performed in the application.

These are the various scenarios which were coded and tested using TCP and UDP and implemented over LAN:

- Case 1: Server-client communication where client chooses song stored locally in the server to play using TCP.
- Case 2: Server-client communication where client sends audio file to server to play using TCP.

- Case 3: Server-client communication where client chooses song stored locally in the server using UDP.
- Case 4: Server-client communication where client sends audio file to server to play using UDP.
- Case 5: Server-multi client communication where one client chooses a song stored locally in the server and all the other clients and this song is played on all the client machines. Done using TCP.

This socket programming was done in Java and multiple computers were used to play the role of the server and clients. The basic methodology behind TCP socket programming in Java is first establishing a socket with a TCP port number. A socket connection means the two machines have information about each other's network location (IP Address) and TCP port. The *java.net.Socket* class represents a Socket. To open a socket, following is used:

Socket socket = new Socket (IPAddress, portNumber)

The client is able to connect to the server using its IP address if the port number used by both are the same and if the server is ready to accept connections. For the communication, streams are used to input and output data. Then the connection is closed.

The methodology behind UDP socket programming in Java is similar to the one above but the main difference is the use of datagrams. First the socket is created and the server accepts connections from clients on the same port, this socket is called a datagram socket:

DatagramSocket socket = new DatagramSocket()

Then datagrams are created for the sending and receiving of data using the datagram socket. These datagrams have the information of the port number and the IP address which are being used in their communication. The *java.net.DatagramSocket* class represents the datagram socket:

DatagramPacket sendPacket = new DatagramPacket(buffer[], bufferLength, IPAddress, portNumber)

Then the sending and receiving of the datagram packets occur over this socket which is then closed after the communication is complete. For playing the music after the transfer using java, a library called BasicPlayer is used [24].

Threads and thread programming was also implemented in Java to increase the number of clients communicating with the server to check the reliability of TCP for efficient communication with multiple clients. For capturing the data and analyzing the performance of the two protocols, Wireshark has been used to capture the packets whenever the code was implemented and data was sent over the network (for all the different cases). The various tools in the software enabled us to analyze the variables such as RTT, latency and packet loss graphically. This data gave a clear picture as to how efficient the protocols and allowed us to choose the one which provided better results for real-time audio transmission for remote music collaboration.

B. Predicting Music Using RNNs

Three models were trained using the Google AI environment Magenta. Magenta is an open-source project exploring the role of machine learning as a tool in the creative process. The three models trained using Magenta's Melody_RNN are basic RNN, lookback RNN and attention RNN. The main aim of this step was to see if melody predictions generated by ML are good enough substitutes for any missing bars of music which might have been unsuccessfully transmitted from one location to the other during real-time music collaboration.

A recurrent neural network (RNN) is a type of artificial neural networks. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. The decision a recurrent net reaches at time step t-1 affects the decision it takes at time step t. Hence, these networks have two sources on input: the present and the recent past.

Long Short Term Memory networks (LSTMs) are a special kind of RNN, capable of learning long-term dependencies. LSTMs enable RNNs to remember their inputs over a long period of time. This is because LSTMs contain their information in a memory which is much like the memory of a computer because the LSTM can read, write and delete information from its memory. This memory can be seen as a gated cell, where gated means that the cell decides whether or not to store or delete information (e.g. if it opens the gates or not), based on the importance it assigns to the information. The assigning of importance happens through weights, which are also learned by the algorithm. This simply means that it learns over time which information is important and which not.

The basic RNN model is a simple LSTM model with one-hot encoding to represent the melodies during training. The input is the one-hot vector of previous event and the label is the target next event. The possible events for this are: note-on, note-off and no event. Also, all the melodies are transposed to MIDI pitch range (48, 84) and the outputs are also in the same range.

The lookback RNN model has custom inputs that allow the model to recognize patterns related to an events position within the measure. In terms of music, a bar is one small segment that holds a certain number of beats. The input vector has information from 1 or 2 bars ago, whether the last event was repeating in 1 or 2 bars ago and also the current position within the measure. The custom labels reduce the amount of information that the RNN's cell state has to remember by allowing the model to more easily repeat events from 1 bar ago or from 2 bars ago. When the labels for training data are being created, if the current event in the melody is repeating the same event from 2 bars ago, the label for that step is set to be 'repeat 2 bars ago'. If it's not repeating the event from 2 bars ago, it checks if it's repeating the event from 1 bar ago, and if so, the label for that step is set to be 'repeat 1 bar ago'.

Only when the melody isn't repeating 1 or 2 bars ago is the label for that step made to be a specific melody event.

This results in melodies that have less variation and have a more musical structure [25].

For the third model, attention RNN is used. Attention is one of the ways that models can access previous information without having to store it in the RNN cell's state. This is done by looking at the previous output generated while generating the current step. Vectors and matrices are the learnable parameters of the model. A length vector with a value for each of the previous step is calculated and normalized using softmax to create a vector or mask values.

These mask values are then multiplied to RNN outputs from previous steps and then summed together. Hence each output contributes differently to the present one. It gets easier for the model even with the long term dependencies since the attention calculation done helps it from not having to store all the past data. Figure 1 describes the steps for training the models:

- The MIDI dataset used for training is obtained online [26] and a subset of it containing classical piano, guitar, and violin MIDI files from various composers is used. There are about 11,000 MIDI files which are used to train the models.
- Moving on to the actual training according to the Magenta environment, first the MIDI files are all converted into NoteSequences which are protocol buffers which are faster and easier to work with than MIDI files. A TFRecord file of NoteSequences was generated.
- These NoteSequences are then converted into SequenceExamples which contain a sequence of inputs and a sequence of labels that represent each melody. The models are then trained one by one using the sequence example files. The number of training steps used at 10,000 for basic and lookback RNN and 8000 for attention RNN. For all the three, a batch size of 64 is used since having a small batch size helps reduce the memory usage, and two RNN layers of size 64 are used to speed up the training.
- One the models are trained, they're used to generate 10 MIDI melody files using a primer MIDI of the song 'River Flows in You' by Yiruma which is of 5 bars. What the model produces is 3 bars of melody following the 5 bars of the primer MIDI to produce a melody of total 8 bars as the inputted number of steps required from the model is 128 steps or 8 bars.

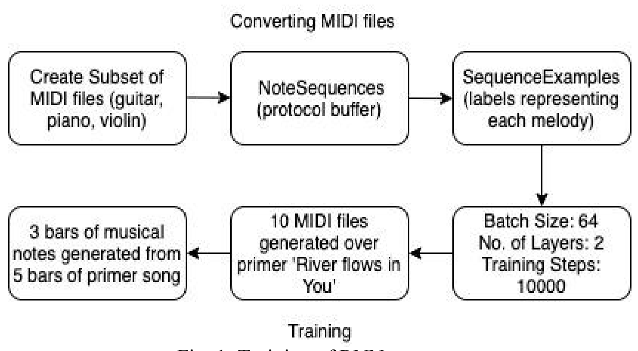

Converting MIDI files
Create Subset of MIDI files (guitar, piano, violin)
NoteSequences (protocol buffer)
SequenceExamples (labels representing each melody)
Batch Size: 64 No. of Layers: 2 Training Steps: 10000
10 MIDI files generated over primer 'River flows in You'
3 bars of musical notes generated from 5 bars of primer song
Training

Fig. 1. Training of RNNs

Hence, the 3 missing bars were simulated on the 5 bar song using the models trained on the testing data. Whenever there's a lag or a glitch, instead of making the musicians stop, the model can predict an approximate of what melody could have been transmitted. This glitch will hence be covered up by machine learning and the receiver will not get to know about it, even the sender will be able to continue playing without worrying about the missed notes. Hence it'll keep the music going.

In this work, the model could only be trained for generation of melodies and not for evaluation or testing of those generated melodies. The evaluation was done by humans who had basic knowledge about music and who had some experience working as musicians. Hence, they were able to rate and distinguish from the 10 MIDI files each model created. It is possible to create an evaluation system using the techniques of ML and can be implemented in the future to reduce the human effort.

IV. Results and Discussions

A. Networking Using TCP and UDP

The following graphs in Figure 2, 3, 4 and 5 and other such graphs were obtained from the packets captured using Wireshark. Analysis was done using the following parameters:

- Packet loss: It is the failure of one or more transmitted packets to arrive at their destination.
- Latency: This is the term used to indicate any sort of delay that happens in data communication.

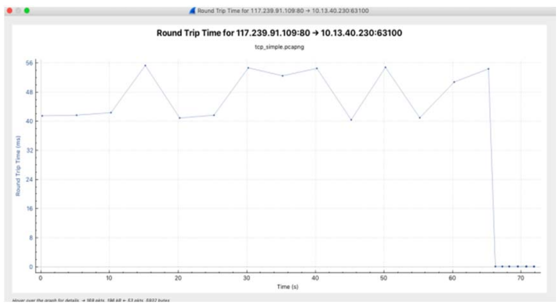


Fig. 2. RTT for Case 1 (Time (s) vs. RTT (ms))

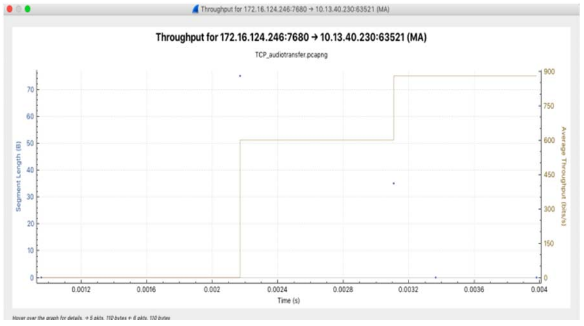


Fig. 3. Throughput for Case 2 (Time (s) vs. Average Throughput (bits/s))

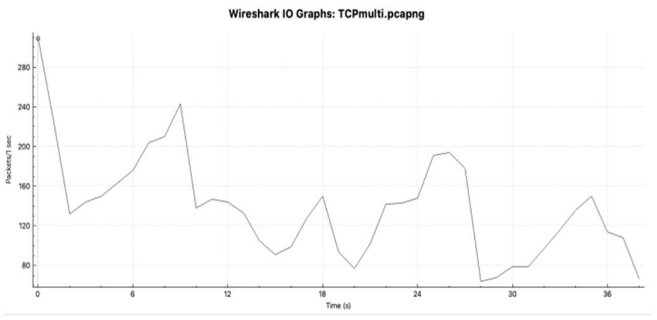


Fig. 4. Latency for Case 5 (Time (s) vs. Packet (per sec))

- RTT: Round-trip time is the time required for a signal pulse or packet to travel from a specific source to a specific destination and come back again.
- Throughput: It is a measure of how many units of information a system can process in a given time.

It was found that it was easier to capture and analyze packets for the TCP cases rather than the UDP cases; UDP isn’t affected by latency and Wireshark doesn't provide the tools to examine the other factors for the same, hence, we focused on the TCP cases.

From the graphs, there is no packet loss in TCP and the values for RTT and throughput also seem sufficient, no packet loss is observed. For case 1, the maximum RTT achieved is 55 ms and fluctuates; in case 2, when audio files are being transferred, the maximum RTT obtained reduces to 0.09 ms, however, the fluctuations for the same reduce. When the number of clients are increased, we get maximum RTT of 0.06 ms. The maximum latency observed goes from 8 packets/sec in case 1 to about 290 packets/sec in case 5. Also, the throughput increases from 2100 bits/sec to 80,000 bits/sec from the 1st to 5th case.

These delays seen can be improved upon by using further technologies and even the RTT can be reduced further since it decreases upon adding clients or sending heavy files and the latency also increases. Hence with further optimization and data handling, this latency can be reduced and the RTT and throughput can be improved.

B. Predicting Music Using RNNs

TensorBoard is used to examine the different trained models. The machine used to train it was a MacBook Air with 8GB RAM and 1.6GHz dual-core Intel Core i5 processor. All the graphs obtained for the three models are

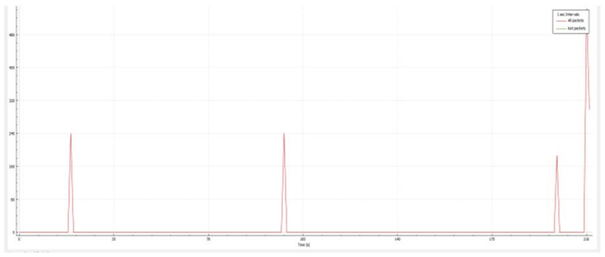


Fig. 5. Packet Loss and All packets (Red) for Case 1 (Time (s) vs. Packet (per sec))

of the form given in Figure 6. They are examined and compared together at the end of this section in Table I. Figure 7 shows the MIDI files generated by the models along with the kind of log-likelihood values of each generated MIDI file by the models in Figure 8. Comparing the three models:

- Of the three models, basic RNN gave results which either had off-scale notes, lots of repetitive notes or empty gaps in between. Whereas attention and lookback RNN provided melodies which sounded better and were closer to the scale of the primer MIDI song.
- The values for the loss, accuracy and perplexity for all of the three are almost the same, hence numerically there’s not much difference in the three models. Even though the accuracy is the same, the outputs for the lookback and attention models is easier on the ear and were preferred by a group of real-musicians.

TABLE I. VALUES OBTAINED BY TENSORBOARD FOR 3 MODELS

	Min. Loss	Max. Accuracy	Min. Perplexity	Range of log-likelihood values
Basic	0.77	0.7315	2.5	-260.5 to -171.3
Lookback	0.91	0.7321	2.55	-272.4 to -186.1
Attention	0.99	0.7131	2.71	-280.4 to -232.9

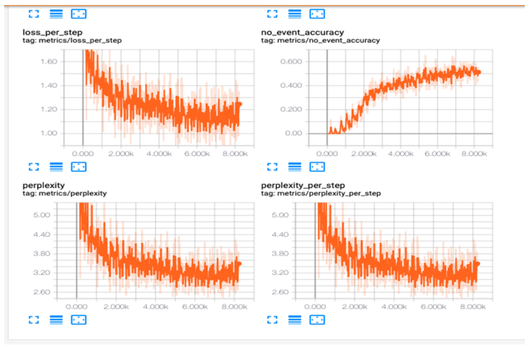


Fig. 6. TensorBoard graphs for models

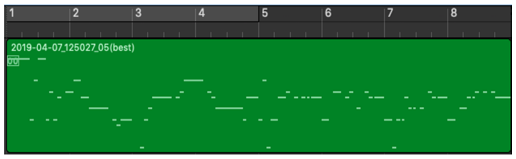
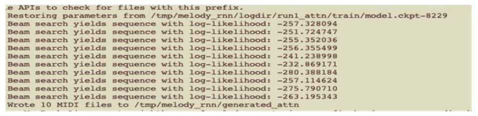


Fig. 7. MIDI files generated by models



```
.e APIs to check for files with this prefix.
Restoring parameters from /tmp/melody_rnn/logdir/run1_attn/train/model.ckpt-8229
Beam search yields sequence with log-likelihood: -257.328094
Beam search yields sequence with log-likelihood: -251.724747
Beam search yields sequence with log-likelihood: -255.352036
Beam search yields sequence with log-likelihood: -256.355499
Beam search yields sequence with log-likelihood: -241.238998
Beam search yields sequence with log-likelihood: -232.869171
Beam search yields sequence with log-likelihood: -280.388184
Beam search yields sequence with log-likelihood: -257.114624
Beam search yields sequence with log-likelihood: -275.790710
Beam search yields sequence with log-likelihood: -263.195343
Wrote 10 MIDI files to /tmp/melody_rnn/generated_attn
```

Fig. 8. Log-likelihood values

The log-likelihood values of the three models are close in ranges of their values; however, basic and lookback RNN have a range of about 90 whereas attention has the range of 48 which is almost half of the other two which runs in parallel with the 'attention' concept behind it.

V. Conclusion and Future Scope

It is found that TCP is a better protocol since UDP protocol didn't provide enough data regarding RTT, latency, packet loss and throughput. It has been observed that UDP is usually preferred for real time situations since it doesn't spend extra time in error correction or in acknowledging the accepted packets. However, as mentioned, the UDP cases taken in this research projects did not give satisfactory results and quantitative values when analyzed with Wireshark. Hence, TCP is the chosen protocol in this study. However, further testing on UDP on larger data files is to be carried out to make a more accurate comparison between the two.

It is also found that machine learning and RNN provides decent generated melodies which can be used to substitute for any lost/corrupted music while communication and hence help in a continuously flowing music collaboration across different geographical locations.

While basic RNN gave very repetitive results, as well as a few MIDI results with gaps, attention and lookback RNN gave significantly more pleasant and more musical melodies which sounded better. Numerically, the training accuracy for basic RNN is maximum and its loss is minimum. However, there's a very miniscule difference between these metrics of the three models, hence according to experimental data and evaluation of the generated music by real musicians, attention and lookback RNN are better than basic.

Efficient models for polyphonic melodies for more diverse and larger datasets can be proposed in the future. Also, the problem of network delay and synchronization needs to be addressed especially when the musical devices are very far in the network. The TCP and UDP protocols need to be tested in extreme situations for this particular use-case. Another future application can be of using IoT to configure musical instruments that can work as the end-points for music collaboration.

References

[1] C. Alexandraki, I. Kalantzis (2007). Requirements and Application Scenarios in the Context of Network Based Music Collaboration. Presented at 3rd AXMEDIS Conf. [Online]. Available: https://www.researchgate.net/publication/255670646_Requirements_and_Application_Scenarios_in_the_Context_of_Network_Based_Music_Collaboration

[2] Method and apparatus for remote real time collaborative music performance and recording thereof, by W. G. Redmann (2005). US7518051B2. [Online]. Available: https://patents.google.com/patent/US7518051B2/en

[3] *Ohm studio*. (2013). Accessed: August 2018. [Online]. Available: https://www.ohmstudio.com

[4] N. Bryan-Kinn and P.G.T. Healey, "Daisyphone: Support for Remote Music Collaboration" in *Proc. NIME*, 2004, pp. 27-30.

[5] C. Chafe, "Distributed Internet Reverberation for Audio Collaboration" in *Proc. AES 24th Int. Conf.*, Banff, 2003.

[6] C. Chafe, S. Wilson and D. Walling, "Physical model synthesis with application to Internet acoustics" in *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Orlando, FL, 2002, pp. IV-4056-IV-4059, doi: 10.1109/ICASSP.2002.5745548.

[7] Juan-Pablo Caceres and A. Renuad, "Playing the Network: The Use of Time Delays As Musical Devices" in *Proc. ICMC*, 2008. [Online]. Available:https://ccrma.stanford.edu/groups/soundwire/publications/papers/caceres_renaud-ICMC2008.pdf

[8] C. Chafe, "Tapping into the Internet as an Acoustic/Musical Medium" in *Journal of The Acoustical Society of America*, Nov. 2007, doi: 10.1121/1.2943002

[9] Juan-Pablo Caceres, "Jacktrip: Under the Hood of an Engine for Network Audio" in *Journal of New Music Research,* 2010, pp. 183-187, doi: 10.1080/09298215.2010.481361.

[10] C. Payne. "Clara: Generating Polyphonic and Multi-Instrument Music Using an AWD-LSTM Architecture". ChristineMcleavey.com. http://christinemcleavey.com/files/clara-musical-lstm.pdf (accessed Sept., 2018)

[11] A. Robers, J. Engel, C. Raffel, C. Hawthrone, D. Eck, "A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music", 2018. [Online]. Available: https://arxiv.org/pdf/1803.05428.pdf

[12] A. Ycart and E. Benetos. (2017). A Study on LSTM Networks for Polyphonic Music Sequence Modelling. Presented at the 18th Int. Society for Music Info. Retreivel Conf., China. [Online]. Available: https://ismir2017.smcnus.org/wp-content/uploads/2017/10/60_Paper.pdf

[13] D. Eck and J. Lapamle. "Learning Musical Structure Directly from Sequences of Music". [Online]. Available:http://www.iro.umontreal.ca/~pift6080/H08/documents/papers/lstm_music.pdf

[14] D. Kang, J. Y. Kim and S. Ringdahl, "Project milestone: Generating music with Machine Learning", Stanford University, CA, USA. Accessed: Aug. 2018. [Online]. Available: http://cs229.stanford.edu/proj2018/report/18.pdf

[15] X. Zhou and A. Lerch. (2015). Chord Detection Using Deep Learning. Presented at Int. Conf. on Music Info. Retrieval. [Online]. Available: http://ismir2015.uma.es/articles/96_Paper.pdf

[16] Heng-Tze Cheng, Yi-Hsuan Yang, Yu-Ching Lin, I-Bin Liao and H. H. Chen, "Automatic Chord Recognition for Music Classification and Retrieval" in *Proc. IEEE International Conference on Multimedia and Expo,* Hannover, 2008, pp. 1505-1508, doi: 10.1109/ICME.2008.4607732.

[17] A. Vaswani *et al.*, "Attention Is All You Need", 2017. [Online]. Available: https://arxiv.org/pdf/1706.03762.pdf

[18] T. Carsault, J. Nika and P. Esling. "Multi-step generation of chord progressions for inference in jazz through recurrent neural networks", 2018. [Online]. Available: https://esling.github.io/documents/mlProj_carsault.pdf

[19] D. Johnson. "Composing Music with Recurrent Neural Network". Hexahedria.com. http://www.hexahedria.com/2015/08/03/composing-music-with-recurrent-neural-networks/ (accessed Sept. 2018)

[20] Cheng-Zhi A. Huang and T. Cooijmans, "Counterpoint by Convolution", 2017. [Online]. Available: https://arxiv.org/abs/1903.07227

[21] *Wireshark: Packet Analyser.* (2018) Accessed: September 2018. [Online]. Available: https://www.wireshark.org/

[22] *NetBeans IDE*. (2018), Apache. Accessed: August 2018. [Online]. Available: https://netbeans.org/

[23] *Google Magenta*. (2018), Google. Accessed: December 2018. [Online]. Available: https://magenta.tensorflow.org/

[24] *Javazoom API*. (2006). Accessed: September 2018. [Online]. Available: http://www.javazoom.net/jlgui/api.html

[25] E. Waite. "Generating Long Term Structures in Songs and Stories". Magenta.Tensorflow.org.https://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn/

[26] *The Largest MIDI collection on the internet,* Reddit, December 2018. [Online].Available:https://www.reddit.com/r/datasets/comments/3akhxy/the_largest_midi_collection_on_the_internet