Vibe.io, Design and Implementation of a Bidirectional Real-Time Mobile Client to Digital Audio Workstation Interface

SHAN JIANG, Georgia Institute of Technology, USA

CCS Concepts: • **Networked Music Performance**;

ACM Reference Format:
Shan Jiang. 2021. Vibe.io, Design and Implementation of a Bidirectional Real-Time Mobile Client to Digital Audio Workstation Interface. 1, 1 (December 2021), 4 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Networked Music Performance is a subfield of Music Technology that investigates how musicians in different parts of the physical space can synchronize their performances online.[Rottondi et al. 2016]

It has been unheard of for a long time, but in recent years it has regained the attention of the industry. On the one hand, the development of network communication technology has made low-latency audio transmission possible, and on the other hand, many emerging needs from the music market are gradually being unleashed.

First, the traditional music recording industry has declined. Since the introduction of mp3, record sales have been under the pressure of mp3 and have been declining year by year, so in recent years only professional musicians and senior music lovers are willing to pay for it.[Liebowitz 2004] What's more, in recent years, the revenue of online streaming media such as Spotify has also gradually slowed down.[Lozic 2020] The traditional music production and distribution process, which is dominated by professional producers, seems to be under new challenges because of its unique cost constraints.

Second, the mobile Karaoke craze continues to grow, with a proliferation of well-known civilian podcasts and low-cost music memes making their way onto social networks.[Viñuela 2022] The sales of the over-the-top microphone with its speaker are overwhelming on online shopping platforms, followed by a slew of USB microphones for smartphones. The sales of these microphones have created an order of magnitude advantage over microphones suitable for professional recording.

The significant changes in the market have naturally triggered an upgrade from hardware to software[Aylett et al. 2013]. For example, in the hardware field, such as Rode launched a series of consumer wireless products to lead the wireless microphone change. Many consumer-level audio interface adapted to mobile devices are also gradually entering millions of homes.

Author's address: Shan Jiang, sjiang340@gatech.edu, Georgia Institute of Technology, Atlanta, Georgia, USA, 30332.

However, no matter how the hardware quality is improved, and no matter how advanced the mobile phone sound effect algorithm is, music will always be the carrier of human emotional communication. For singers, it is difficult to capture true emotions without a group of listening audiences, and it is difficult to learn advanced techniques without a professional and responsible instructor. Also, it is difficult to go further and enter the world of professional music.

The Internet provides a platform where part-time sound engineers can provide real-time accompaniment and mixing services to entry-level singers from home. Singers no longer have to worry about finding the right accompaniment, and can also get professional singing advice and real-time mixing from sound engineers. These activities are no longer limited to time and venue. Recording and singing again and again will become more like a light-hearted music party than a work against time.

Vibe.io was developed with this goal, creating a bridge between the phone and the PC, and transferring data between the phone software and the DAW via WebRTC. This allows the user to easily communicate in real-time with a remote mixer, the mixer can also mix for the user in real-time.

2 RELATED WORK

2.1 Jacktrip

This is a top network music interface in the network music performance field at present. It uses a bit drop or interpolates method to reduce the delay overhead caused by network jitter[Cáceres and Chafe 2010] Currently, the Jacktrip Foundation has started several projects, including Jacktrip virtual studio and Jacktrip radio.

The limitation of Jacktrip is that it focuses on the audio bit stream rather than the network package. It does not provide a method to break through the firewall and public network IP restrictions, so it requires a more demanding network environment. Based on the Raspberry Pi, Jacktrip has also developed an embedded network audio streamer, but the price of $200 and the complicated construction method still block the enthusiasm of most users.

2.2 Sonobus/Jamulus

Unlike Jacktrip, both Sonobus and Jamulus[Fischer 2015][Mall and Kilian 2021] provide software applications that are easier to install. Users can easily connect to the cloud server through the network, and further communicate with remote collaborators.

However, their audio performance isn't quite as good. This is because their audio stream is being transported via UDP packages on the Internet, and the fluctuation of package flow causes a significant amount of package loss that causes random audio drops.

Adding a buffer on the receiver's side helps to address this issue, yet it generates more latency. In our application, for the specific music recording case, we apply a technique to compensate for this latency to the receiver's monitor channel to align both the remote and local track. Details are introduced in the next section.

2.3 Smule

This is a mobile phone karaoke software, users can find their favorite songs on their mobile phones, record their own singing, and add sound effects.[Wang et al. 2009] However, Smule may not be able to provide a rich enough music library accompaniment due to copyright issues. Moreover, as mentioned earlier, mobile phones at this stage cannot replace sound engineers to provide singers with real-time accompaniment and high-quality mixing services.

2.4 Soundtrap

This is a collaborative Digital Audio Workstation.[Lind and MacPherson 2017] Musicians and sound engineers can collaborate on recording and editing audio tracks on the web. However, the collaboration of users must be achieved through constant web page refresh, and the synchronization problem is difficult to solve.

3 DESIGN CONCEPT

Jacktrip, Sonobus, and Jamulus are focusing on the real-time coop between musicians, it requires an extremely low-latency environment. As discussed above, Jacktrip established an intricate audio restoration method while falling short of being compatible with modern internet porting techniques that decreased the usability for entry-level users. Sonobus and Jamulus sacrificed the audio quality to enhance their low-latency service. There seems no perfect solution for the absolute low-latency music transmission over the internet yet.

However, online karaoke does not need an absolute real-time environment. In this case, the engineer plays a background music track in their DAW, this track is sent to the singer, then the singer sings along to this track, and his or her vocal track is sent back to the engineer's DAW. Thus, the engineer can further mix the song and advise the singer.

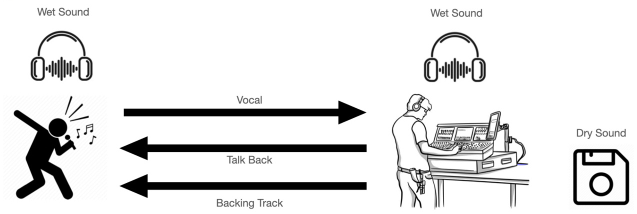


Fig. 1. A Typical Case for Music Recording

In order to do this, our application must provide these services: A bi-directional high-quality audio channel over the internet with relatively low latency. A set of vocal effects is applied to the singer's monitor channel, and the engineer should be able to tweak these effects remotely. The latency compensation to synchronize both background and vocal tracks in the engineer's local monitor channel.

This software consists of four parts, React and Web Audio API is responsible for front-end effect rendering and control behavior mode, WebRTC is responsible for high-quality network audio communication and the VST Plugin is responsible for latency compensation in the DAW. In the WebRTC module, the STUN server uses the google open source service, the Firestore database is responsible for the information exchange between ICE and SDP, and the WebRTC Datachannel is responsible for the control linkage between peer A and peer B. In the VST Plugin, a trigger signal for delay detection will be sent out and then monitored to calculate to add a corresponding delay. In Web Audio API modules, audio nodes provide regular sound effects and handle the channels mixing & routing.

In the singing mode, the recording engineer on the B side plays the music accompaniment in the DAW, and the music is transmitted to the headphone of the singer on the A side through WebRTC. After the singer hears the accompaniment, he or she sings the song and the vocal returns to the B side from the WebRTC channel. At this point, the sound engineer will be able to hear the vocal track in sync with the accompaniment music, thanks to the VST plug-in that provides dynamic delay compensation on the accompaniment track.

In playback mode, the recording engineer on the B side plays both the recorded accompaniment and vocals, and the music is sent back to the singer's headphones through WebRTC, and the singer can hear the playback.

In addition, singers in singing mode can hear the real-time monitor sound effects provided by the web audio module in the local browser. If the singer is not satisfied with the current effect, the recording engineer can make adjustments from the B side.

4 IMPLEMENTATION

4.1 Architechture

As discussed in the previous section, the whole system is constituted of three main sections: the RTC communication channel, the Peer A Web APP and the Peer B AU/VST plugin.

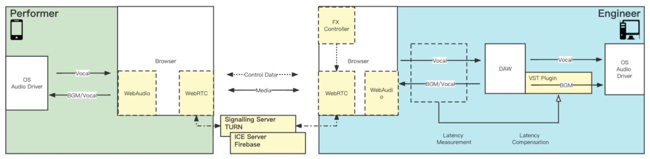


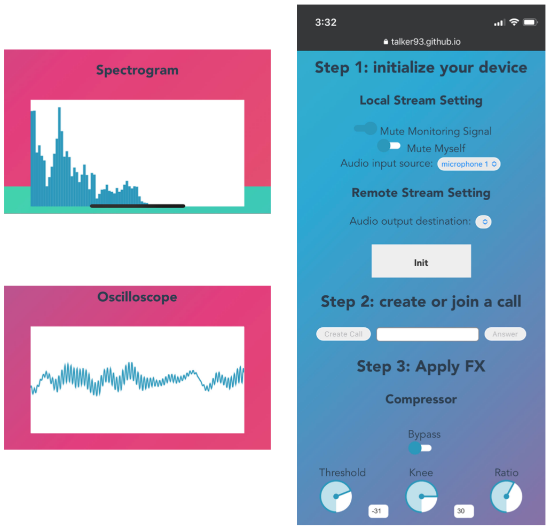
Fig. 2. An Overall Structure

4.2 Peer A - Web APP

This web app comes with React JS and NexusUI library to support its frontend GUI as a single-page application running in the browser. React enables a manageable time sequence of time-sensitive interactions, such as IO selection, Web Audio Nodes loading, etc. NexusUI provides a bundle of audio GUI components, such as knobs, sliders, etc.

Besides, a set of Web Audio API nodes has been implemented into this app. It includes some general vocal FX, such as Compressor, Convolver, Reverb, etc.

Lastly, a background process is running to listen to the change of some specific data in the database, which will be modified when the remote engineer sends the update query of vocal FX parameters each time.



Step 1: initialize your device

Local Stream Setting

Mute Monitoring Signal

Mute Myself

Audio input source: microphone 1

Remote Stream Setting

Audio output destination:

Step 2: create or join a call

Step 3: Apply FX

Compressor

Bypass

Fig. 3. Vibe.io - Web APP

4.3 Peer B - AU/VST plugin

JUCE is being used as the main framework for this plugin. It provides not only the GUI framework but also a series of classes of managing audio buffers and packaging them with an AU/VST standard interface.

In order to track the loop-back latency, the plugin utilizes the getStat() method in the WebRTC C++ standard library, which acquires the RTCStat.timestamp of each audio package.

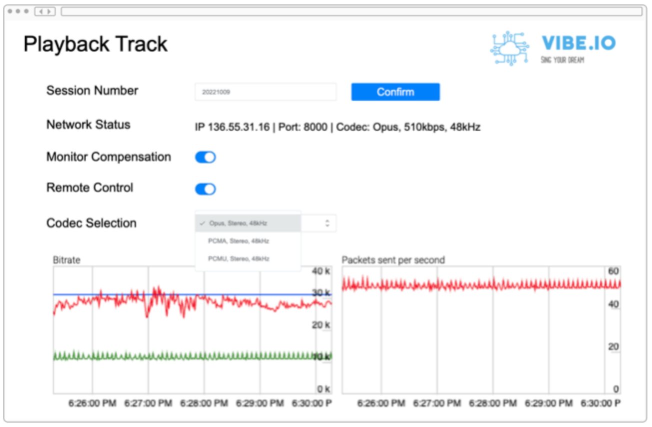
Other than that, the current official WebRTC C++ library is inclined to the development of network browsers, such as Chromium, which does not fit well in this project. Vibe.io - AU/VST plugin utilizes a third-party SDK provided by Dolby.io. However, the Dolby.io WebRTC C++ SDK is also in its early time, some API has not yet been properly packaged. Thus, the plugin implements a socket channel for this inter-process communication between Dolby.io SDK and the JUCE core process.

5 TECHNIQUE SELECTION

5.1 Native or Web

For the convenience of discussion, the singer's end and the sound engineer's end are referred to as peer A and peer B for short. For peer A, we need to consider whether to base the development on a native application or on the web.

The advantage of being based on native applications is that the software performance and stability are better, but the compatibility is also reduced. Each native application can only be developed on one platform. The development environments of iOS and Android are quite different. Parallel development is almost equivalent to making two different software. In addition, the development difficulty of native applications is also significantly higher than that of web



Playback Track

Session Number 20221009

Network Status IP 136.55.31.16 | Port: 8000 | Codec: Opus, 510kbps, 48kHz

Monitor Compensation

Remote Control

Codec Selection ✓ Opus, Stereo, 48kHz

PCMA, Stereo, 48kHz

PCMU, Stereo, 48kHz

Fig. 4. Vibe.io - AU/VST Plugin

development. Whether you choose to react native or object-c, it involves a large number of native system API calls and a large number of low-level memory access commands. These processes will undoubtedly greatly increase the time required to develop and debug.

On the contrary, web-based development has better compatibility, but the stability will also be greatly reduced. The same mobile phone, using different browsers, or even using different versions of the same browser, will cause some API failures or exceptions. At the same time, the operation of the browser takes up a lot of memory, and it is subject to many restrictions of the operating system in terms of file read and write permissions, which will greatly limit the functions and performance of the software. On the whole, since we tend to develop a more usable version in the shortest time and strictly limit the usage scenarios, web-based development can basically realize the required product functions, and at the same time can greatly speed up the development progress. From this, I finally chose the web as the primary development method.

5.2 WebRTC or RTMP

RTMP is in fact a more classic and popular network streaming protocol. Developed by Adobe Macromedia, this protocol enables chunked transmission of audio streams. At the same time, there are many mature API interface solutions. However, the transmission of RTMP is based on the TCP protocol, thus it is difficult to achieve low-latency transmission, which is acceptable in one-way transmission, but has strong limitations for the more interactive two-way channel.

WebRTC is a streaming media transmission protocol led by Google in recent years. Its biggest feature is that it does not require a media server to forward audio data, and each user on the node can easily establish P2P remote communication. At the same time, WebRTC supports many advanced streaming media encoding formats, including Opus, so to transmit high-quality audio streams at extremely low bit rates.

However, compared to RTMP, WebRTC also has a big disadvantage. It mainly focuses on serving browsers rather than native applications. Therefore, using WebRTC in native applications, such as WebRTC-JAVA or WebRTC-C++, may require the use of many immature APIs. I have also considered some recently popular commercial solutions. For example, Agora wraps WebRTC in the native language of each platform and calls the agora API during development. However, after careful study, it is found that the applicable scenarios of these solutions are limited to conventional application requirements. For example, in the choice of audio codec format, there is often no corresponding encapsulated API.

On the whole, WebRTC is more suitable for web applications, while RTMP is more suitable for native applications. Our project is based on web development, so choosing WebRTC should be a reasonable choice.

5.3 WebRTC Datachannel or Firebase relay

In order to develop a remote effect controller, we need to consider the forwarding process of these control signals.

In the original scenario, I used Firebase store-and-forward data. All sound effect setting parameters are stored in the database, and peer A and peer B keep monitoring the database at the same time. When one end initiates a data update, the other end will also query the corresponding data and update it to the local settings.

This solution is easier to implement than WebRTC Datachannel, but it has huge risks. For effect parameters, such as the Gain value, one knob may be updated several times in a second, which means that the other end also needs to be accessed several times in real-time. This will greatly increase the amount of access to the database. I have also tried to manually set the delayed averaging method, but the amount of access is still huge, even triggering the malicious access protection mechanism of the database.

In the end, I chose the Datachannel API provided by WebRTC. The advantage of this solution is that there is no third-party forwarding for point-to-point real-time communication, so the data update logic is easier to design, and there is no need to worry about the access volume of the database.

6 CHALLENGE

The reason that a seemingly plausible application hasn't been created is often related to its complexity. As the cutting-edge technology in the streaming area, most native WebRTC implementations remain to be the intellectual property of companies and haven't been open-sourced. Moreover, applying dynamic latency compensation requires a set of thorough strategies to deal with unexpected scenarios, such as a frequently varied delay.

Nonetheless, there are backup ways to circumvent these potential risks. Here are the solutions below:

(1) Challenge 1: Dynamic latency compensation may not achieve the desired effect.
  - Solution A: Add a long-enough buffer for latency compensation. Unsurprisingly, this solution deals with a dramatically changed delay, with a second-length buffer, any variation won't surpass this duration. Its downside is still can't deal with the transition between varied clips at their onset and end position.
  - Solution B: A backup full-packaged audio file. A file can be created in the local file system when the ending flag in the audio is received. The engineer can manually drag that clip into his or her DAW to the right position. It avoids audio latency and quality problems.

  Specifically, this file can be transported through either WebRTC in the synchronous UDP method or other TCP asynchronous methods, such as a web files management service. In our practice, we will first try solution A, then goes to solution B if A is not workable. In the best case, we can provide both solutions A and B together.

(2) Challenge 2: Few ready-to-use WebRTC Native C++ resources To deal with this challenge, this project has to consider the utilization of some non-open-sourced services. There are options, such as Mediasoup or Agora, but they are dedicated to mobile WebRTC native development instead of desktop. This project finally choose DolbyIO. So far DolbyIO works well with this project.

7 CONCLUSION AND FUTURE WORK

In summary, the development work for this semester focuses on the following points:

(1) The realization of WebRTC Native C++ on the desktop platform;
(2) The establishment of peer connection between WebRTC C++ and its web version;
(3) The implementation of injection/recording of audio stream from WebRTC C++;
(4) The inter-process communication between DolbyIO SDK and JUCE core process;

In the future, I will focus on implementing the following modules:

(1) Integrate both DolbyIO SDK and JUCE into the plugin bundle;
(2) Execute base case testing between native and the web end;
(3) Implement the latency tracking & compensation;

REFERENCES

Mathew Aylett, Yolanda Vazquez-Alvarez, and Lynne Baillie. 2013. Evaluating speech synthesis in a mobile context: Audio presentation of Facebook, Twitter and RSS. In *Proceedings of the ITI 2013 35th International Conference on Information Technology Interfaces*. IEEE, 167–172.

Juan-Pablo Cáceres and Chris Chafe. 2010. JackTrip: Under the hood of an engine for network audio. *Journal of New Music Research* 39, 3 (2010), 183–187.

Volker Fischer. 2015. Case Study: Performing Band Rehearsals on the Internet With Jamulus. *URL: https://jamulus. io/PerformingBandRehearsalsontheInternetWithJamulus. pdf* (2015).

Stan J Liebowitz. 2004. Will MP3 downloads annihilate the record industry? The evidence so far. In *Intellectual property and Entrepreneurship*. Emerald Group Publishing Limited.

Fredrik Lind and Andrew MacPherson. 2017. Soundtrap: A collaborative music studio with Web Audio. (2017).

Josko Lozic. 2020. Comparison of business models of the streaming platforms Spotify and Netflix. *Economic and Social Development: Book of Proceedings* (2020), 110–119.

Peter Mall and Jürgen Kilian. 2021. Low-Latency Online Music Tool in Practice. (2021).

Cristina Rottondi, Chris Chafe, Claudio Allocchio, and Augusto Sarti. 2016. An overview on networked music performance technologies. *IEEE Access* 4 (2016), 8823–8843.

Eduardo Viñuela. 2022. 'What's So Funny?': Videomemes and the Circulation of Contemporary Music in Social Media. *Contemporary Music Review* 41, 4 (2022), 445–458.

Ge Wang, Georg Essl, Jeff Smith, Spencer Salazar, Perry R Cook, Rob Hamilton, Rebecca Fiebrink, Jonathan Berger, David Zhu, Mattias Ljungstrom, et al. 2009. Smule= Sonic Media: An Intersection of the Mobile, Musical, and Social.. In *ICMC*.