Mobile Technologies for Music Interaction

Antonio Deusany de Carvalho Junior

Thesis presented
to
Institute of Mathematics and Statistics
from
University of São Paulo
in
partial fulfillment
for the
requirements of the degree
of
Doctor of Science

Program: Graduate School in Computer Science

Advisor: Prof. Dr. Marcelo Gomes de Queiroz

During this work, the author received scholarship from CAPES

São Paulo, May 2017

Mobile Technologies for Music Interaction

This thesis version contains the changes, corrections, and suggestions from the Judging Commission during the defense of the original version that took place on May 24th, 2017. One copy of the original version is available at the Institute of Mathematics and Statistics from University of São Paulo.

Judging Commission:

- Prof. Dr. Marcelo Gomes de Queiroz (advisor) - IME-USP
- Prof. Dr. Alfredo Goldman vel Lejbman - IME-USP
- Prof. Dr. Daniel Macêdo Batista - IME-USP
- Prof. Dr. Georg Essl - University of Wisconsin-Milwaukee
- Prof. Dr. Jason Freeman - Georgia Tech

Agradeço às energias que proporcionam a existência de tudo.

Acknowledgments

Thanks God.

Heartfelt thanks to my Family.

Cheers to my friends “from Jampa, Sampa, and Ann Arbor”, Marcelo Queiroz (d’ Advisor), Georg Essl (d’ Supervisor), USP, UMich, CAPES, RNP, and many others from the universe and beyond.

A scientific expertise recognition to Alfredo Goldman vel Lejbman, Lisandro Zambenedetti Granville, José Luiz Ribeiro Filho, Antônio Carlos Fernandes Nunes, Fábio Rodrigues Ribeiro, Edson Lima Monteiro (aka Boni), Victor Igor de Lima Andrade, Luiz Eduardo Silva dos Santos, Henrique Cabral de Souza Rodrigues, Valter Pereira, Wagner Pereira, Rogerio Herrera Mendonça, Andre Lopes da Silva, Kyle Banas, Roy Hockett, Michael Stanton, Marco Antonio M. Teixeira, William Alexandre Miura Gnann, Alex Moura, Aluizio Abrahao Hazin Filho, Brian Pullin, Brady Farver, Guilherme Ladvocat, Nathan Miller, Christian O’Flaherty, Iara Machado, Alex Soares de Moura, Guillermo Cicileo, Eric Boyd, and many many other professionals.

Fond memories to Tom and Brenda Reedy, Alexia De La Cuba, Marika Feuerstein, Thais Scalia, Panagiotis Trikaliotis, Roque Aleixo, Helena Pereira, Marisa Villas Boas, and Escher House.

A great smile to Luvyn, Lara, Leã, Petit, Neve, Mingau, and Suri.

Namaste to all souls and energies surrounding me.

Thank you all for supporting this research process.
I am thankfully glad you exist.

Abstract

DE CARVALHO JUNIOR, A. D. **Mobile Technologies for Music Interaction**. 2017. 187 p. Thesis (Doctorate) - Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2017.

Mobile music applications are becoming commonplace around the world, and mobile devices are used as digital instruments everywhere. Controlling, performing, or composing music in real time with these devices encourages collaboration and interaction, as telecommunication improvements allow many people to cooperate through local networks or the Internet. In this context, the aim of this thesis is to evaluate mobile technologies that might be suitable for mobile musicians and their audiences while performing or composing. Specifically, the main goal is to explore technologies for collaborative mobile music and to obtain quantitative and qualitative data regarding these technologies and their settings, so that composers might take full advantage of the available options for mobile applications. This evaluation focuses on message exchange using Multicast, Unicast, and Cloud Services, using academic networks as the main pathway. With these services, messages are organized as packet streams, characterized by different sizes and time intervals. Evaluation also includes the development of several applications that make use of these technologies running on Android devices and web browsers. These applications were used in actual performances, serving as both evaluation tools and experimental music instruments. The results were analyzed in terms of round trip time and data loss under very different configuration scenarios, demonstrating that although some obvious impediments are unavoidable (e.g. significant delays in international settings), it is possible to choose the specific technology and achieve interesting results under most music application scenarios. I argue that although in theory Multicast appears to be the best technology to use by far, it is the most difficult to implement due to the burden of configuring every step of the network pathway. On the other hand, Cloud Services are certainly slower than direct connections, but are the most compatible and easiest technology to set up, and are definitely suitable for many collaborative music experiences. To conclude, there is a discussion of how mobile music practitioners can take advantage of these results for composition and performance by considering specific technological advantages or drawbacks that are inherent to each technology and setting.

Keywords: mobile music, computer music, computer networks, cloud services.

Resumo

DE CARVALHO JUNIOR, A. D. **Tecnologias Móveis para Interação Musical**. 2017. 187 f. Tese (Doutorado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2017.

Aplicações de Música Móvel estão se tornando populares ao redor do mundo e dispositivos móveis estão sendo utilizados como instrumentos musicais em diversos lugares. Controlar, apresentar ou compor música em tempo real com estes dispositivos estimula a colaboração e a interação, e os avanços nas telecomunicações permitem a um grande número de pessoas cooperar musicalmente através de redes locais ou da Internet. Neste contexto, o objetivo desta tese é avaliar as tecnologias móveis que podem ser úteis para músicos e público na performance ou na composição. De maneira mais específica, o objetivo principal é explorar as tecnologias para Música Móvel colaborativa e obter resultados quantitativos e qualitativos referentes a estas tecnologias e suas configurações, de modo que compositores possam usufruir de todas as vantagens das opções para aplicações móveis. Esta avaliação enfoca a troca de mensagens através de Multicast, Unicast e Serviços em Nuvem utilizando redes de computadores acadêmicas como principal caminho. Através destes serviços as mensagens foram organizadas como fluxos de pacotes caracterizados por diversos tamanhos e intervalos entre envios. A avaliação também inclui o desenvolvimento de diversas aplicações fazendo uso destas tecnologias para dispositivos Android e navegadores Web, que foram utilizados em performances reais, servindo tanto como ferramentas de avaliação quanto como instrumentos para música experimental. Os dados são analisados com relação ao tempo de ida-e-volta e perda de pacotes em diferentes configurações de cenário, demonstrando que apesar de alguns impedimentos óbvios serem incontornáveis (como o longo atraso em configurações internacionais, por exemplo), é possível escolher tecnologias adequadamente e alcançar resultados interessantes em muitos cenários de aplicações musicais. Argumento que apesar de em teoria o Multicast se apresentar de longe como a melhor tecnologia para estes cenários, ele é o mais difícil de ser implementado devido à grande complexidade na configuração de cada parte da rede para seu uso. Por outro lado, Serviços em Nuvem são certamente mais lentos, porém se apresentam como os mais compatíveis e fáceis de configurar, sendo definitivamente os mais adequados para muitas experiências de música colaborativa. Em conclusão, discuto como profissionais de Música Móvel podem se aproveitar dos resultados apresentados, considerando as vantagens e desvantagens tecnológicas específicas que são inerentes a cada tecnologia ou configuração quando utilizada em performances e composições musicais.

Palavras-chave: música móvel, computação musical, redes de computadores, serviços de nuvens.

Contents

List of Abbreviations

AC1750	TP-Link AC1750 Archer C7 Wireless Dual Band Gigabit Router
ADT	Android Development Tools
bps	Bits per second
CHI	ACM Conference on Human Factors in Computing Systems
CIDR	Classless Inter-Domain Routing
Compmus	Computer Music Research Group - IME — USP
D686	LG D686 G Pro Lite Dual
dBi	Decibels Isotropic
DSP	Digital Sound Processing
ECA	Escola de Comunicação e Artes — USP
IrDA	Infrared Data Association
Gbps	Gigabits per second
I2	Internet2
ICMC	International Computer Music Conference
IEEE	Institute of Electrical and Electronics Engineers
IEEE-SA	IEEE Standards Association
IME	Instituto de Matemática e Estatística — USP
ISM	Industrial, scientific, and medical
kbps	Kilobits per second
LAN	Local Area Network
MANET	Mobile Ad-hoc Network
MIS	*Museu da Imagem e do Som de São Paulo* (São Paulo Museum of Image and Sound)
MobileHCI	ACM International Conference on Human-Computer Interaction with Mobile Devices and Services
Mbps	Megabits per second
MIDI	Musical Instrument Digital Interface
MM	Mobile Music
MoPhO	Mobile Phone Orchestra (Stanford University)
MTU	Maximum Transmission Unit
MVC	Model View Controller
NFC	Near Field Communication
NIME	Conference of New Interfaces for Music Expression
NuSom	Research Centre on Sonology - ECA — USP
OSC	Open Sound Control

OTG	USB On-The-Go
P2P	Peer to peer
PDA	Personal digital assistant
RAD	Rapid Application Development
RNP	Rede Nacional de Ensino e Pesquisa (Brazilian National Research and Education Network)
RoR	Ruby on Rails
RTT	Round Trip Time
S3	Samsung GT-I9300 Galaxy SIII
SLA	Service Layer Agreement
SMC	Sound and Music Computing Conference
ssh	Secure Shell
UFPB	Universidade Federal da Paraíba
UMich	University of Michigan
USP	Universidade de São Paulo
VLAN	Virtual LAN
Z3	Sony D8533 (and D8503) Xperia Z3 Compact

List of Figures

List of Tables

Chapter 1

Introduction

Music, science and technology intersect at many different points and are of interest to many groups, such as music practitioners and researchers, industry and academia. Digital synthesis of sound, for instance, is practically synchronous with the appearance of digital computers, and even though electronic instruments existed at the same time, the diversity and quantity of electronic/digital instruments has grown rapidly since then. Musicians soon started to explore the use of computerized technology for performance and composition, including experimenting with newer technologies in their work as soon as they became available.

Music collaboration and interaction are as old as music itself and take many forms, such as defining a common interpretation of a piece during a group music performance. In most situations, musicians needed to be in the same place and hear (and see) each other to play together and explore performance alternatives. With computer networks and recent telecommunication technologies, the possibility of distributed music playing became feasible, partially replacing the aerial acoustic transmission of sound for electronic signal transmission (audio and other digital data) through telephone lines, Intranet, and the Internet that opened up new the ways in which music collaboration and interaction could take place.

At another front of technological developments, portable devices appeared on the market and entered the music scene. Synthesizers could be devised as minikeyboards running on batteries, and smartphones became hosts for musical instrument apps, based on knowledge and technology inherited from the fields of mobile computing and computer music. Interest in portability considering size, space, and wireless connections accompanied the emergence of music performance groups using mainly mobile controllers and instruments, a practice that is well-described by the term Mobile Music.

The recent boom of musical application development for smartphones using mobile technologies explores the fact that many users are always connected and able to interact with other users. Several music performances connecting participants through local networks and web servers have taken place worldwide with very interesting results. A downside is the fact that the constraints imposed by the systems and technologies used restricts scalability and, in many cases, user participation in distant places is hampered by network infrastructures or server capacities. Another downside is that overcoming the complexity and logistics of implementation of such systems and technologies renders them virtually inaccessible to musicians without technical knowledge (or the assistance of a person with this knowledge).

Facing these two downsides is a challenge that characterizes the purpose of this study. The literature presents many practical instances of performances using systems that allow music collaboration through computer networks, although most of them present many difficulties (and even impossibilities due to lack of detail) for reimplementation, which is bound to be burdensome whenever possible. Moreover, few network evaluations in terms of statistical data were made with these technologies that could help future systems/technologies and music performances be designed building upon previous experiences.

From my personal perspective, it is possible to use mobile and network technologies for music

purposes without a profound knowledge regarding their technical details. Some available technologies, such as Cloud Services and Multicast, can provide connections for music interaction, even in the long-distance scenarios. Users' mobile devices are already communicating with these, but the awareness of their usefulness for music purposes remains unknown to many potentially interested parties, which creates a gap between potential and actual use of these technologies for music collaboration and interaction. Bridging this gap is the motivation leading to the goals of this study.

1.1 Goals

This work explores current technologies for collaborative music, by mapping, development, and experimentation of applications for mobile music interaction through network technologies. This exploration follows two main approaches: using the technologies for application development, and using the network for collaboration and interaction.

The primary goal of this work is to evaluate technologies for mobile interaction. This evaluation requires the study of current state of art in order to propose new solutions, improve past applications, and devise new approaches. The qualitative evaluation of existing technologies is based on studying, implementing, and using them in different musical ways to uncover what prevents interested musicians from taking full advantage of these technologies, by direct observation and by collaborating with composers and performers to learn what their perspectives unveil. The quantitative counterpart study presented in this thesis provides statistical data of experimental tests over different data representations and network settings, which may serve artists as feasibility studies for different scenarios involving music collaboration and interaction in mobile music.

A secondary goal of the research process was to create partnerships with musicians and artists interested in using these technologies. Applications for specific music contexts were developed to illustrate and facilitate the use of these technologies (in terms of coding or technical knowledge) for actual music scenarios. These applications were used in the music performances for which they were developed, and some of them are still being used (by the musicians who co-authored them and also by others).

1.2 Research description

A first step in the research process was to study the relevant literature on mobile music. Many works present the use of computer networks in music making, especially work related to distributed concerts, under the terms network music and internet music. These are described in Section 2.1. Music ensembles using computers and laptops as musical instruments for networked collaborative performances dealt with many technical issues, such as latency, jitter, and packet loss, especially when using non-local networks (although they also had some restrictions when using local networks). Scalability is also an important issue in many projects using home routers or web servers (instead of distributed solutions). Although most projects focused on performing music within a given space (e.g. a theater), some considered the problem of expanding the number of participants to reach for larger audiences. In this last case, the technologies used required technical adjustments and configurations that are unsuitable for users without a technical background, since they require knowledge regarding setting up web servers or virtual machines in the Cloud.

For the experimental part, this project took advantage of an academic network which allows many setup options and offers gigabit interconnection (between specific nodes); its evaluation is described in Chapter 5. Unicast and Multicast[1] are services theoretically available within the networks selected for evaluation in the experiments. Pusher and PubNub Cloud Services were also evaluated in comparison with Unicast and Multicast. These services are unavailable in the academic network, so packets must transit through external nodes. In this case, Cloud Computing benefits were taken

[1] Unicast was evaluated using IPv4 and IPv6 technology, while Multicast was available only through IPv4 at the time of this research.

into account, such as the localization of the clusters and their scalability. All services were evaluated using a loopback method, in which a message is sent back to the sender as soon as it is received by a device with timestamps registering the moments when the message passes through each end of the loop.

An application called *PushLoop* was specifically designed for the quantitative evaluation of the above services. This application is discussed in Section 4.3. Many other applications developed during this research process as part of the technology evaluations and they are presented in Appendix E, including their technical discussion and motivational aspects. The codes for these applications are available at Github[2] under different licenses when defined by the creators.

Both quantitative experiments and music applications are products of this research, which are contributions to the field of mobile music with the intention of helping interested musicians and non-musicians to explore available technologies for mobile music. Most contributions presented here were also published as papers at international conferences in order to help disseminate the knowledge gained during the research process. The published papers are briefly discussed in the Conclusions (Chapter 6) and are included at Appendix F. The mobile music applications developed, as well as the published papers, are an important part of the contributions of this thesis. They were left to the appendices for better textual organization, in order not to disturb the conceptual flow that guided the research process. Their reading is considered indispensable for a thorough appreciation of this thesis, its context and results.

[2] Author's Github account: www.github.com/deusanyjunior (visited on May, 2018)

Chapter 2

Mobile Music Overview

> Let the music in tonight
> Just turn on the music
> Let the music of your life
> Give life back to music.
>
> Daft Punk

The idea of using mobile devices in art emerged with the development of mobile technologies during the 1990's (de Souza e Silva, 2004). The first experiences with mobile music date back to the beginning of the 2000's, when cellphones became mobile, acquired custom ringtones and began to come supplied with long-life batteries. At this time, mobile networks were mostly used for telephone calls, but this was enough to awaken creativity in music. Although the term had first been theorized as "*musica mobilis*" during the 1980's in a discussion regarding the Walkman and its popularity (Gopinath and Stanyek, 2014a, p. 5–6), those recent events characterized the beginning of a new research field called Mobile Music (MM).

Some authors consider the "Dialtones" (Levin, 2001) music concert as the first MM public concert (Wang 2014; Weinberg 2005). At this event, the audience was invited to download a ringtone and sit in an assigned seat in the theater before the concert started. The performer then called specific phone numbers at predetermined times during the performance using a computer program to create the music. This type of concert seems to avoid user interaction, as the audience participates only passively, but it did inspire many other MM experiences henceforward.

In the early 2000's, the popularity of mobile devices was increasing and many music experiences using them were being conceived. Golan Levin, author of Dialtones, decided to register many experiences and experiments from that time in an informal catalog (Levin, 2004). Examples from that list include suspending 1,000 smartphones ("*Handywolke*" by Peter Hrubesch and Dirk Scherkowski) and illuminating helium balloons based on electromagnetic fields affected during calls ("*Sky Ear*" by Usman Haque). The innovative creativity of MM was then publicized to lay audiences through performances and scientific papers.

The computer music research field presents important annual conferences that have become popular venues in which to publish information about new MM experiences. The conference on New Interfaces for Music Expression (NIME) is one of the main events that considers and explores new methods to use mobile devices as interactive instruments or interfaces. Aesthetic and musical aspects of MM are more commonly discussed at the International Computer Music Conference (ICMC), while more computational outlooks are usually presented at the Sound and Music Computing Conference (SMC). Conferences such as the ACM Conference on Human Factors in Computing Systems (CHI) and the ACM International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI) are other venues that explore MM projects outside the computer music area.

Although there are many different conferences, topics overlap among them, and I will partition the discussions in the following thematic axes: interventions at determined spaces; collaborations;

systems for application development; new musical applications; reports; and books. The papers discussed below are selected examples in MM that offer a more or less chronological account of the field on each of these thematic axes.

An early intervention using MM is described by Gaye *et al.* (2003) regarding the Sonic City project. Users were able to create electronic music while walking on the streets with a portable kit that included a small laptop. Tanaka (2004) discussed a system for collaborative music creation using Personal Digital Assistants (PDA) at NIME 2004, which applied the idea of real-time single stream composition. The results of the first mobile phone orchestra, which took advantage of some features of the Nokia N95 smartphone, were presented during the ICMC 2008, and included a review of many past works on MM (Wang *et al*, 2008). At SMC 2009, Koray Tahiroglu presented a system for audience participation using mobile devices and discussed the way in which it extended mobile phone orchestras (Tahiroglu, 2009).

There are several examples in the literature that consider systems for application development and applications made for music creation. At ICMC 2003, a port of Pure Data for the PocketPC platform was presented (Geiger, 2003), while Schiemer and Havryliv (2005) presented a way to convert Pure Data patches to J2ME applications created during the Pocket Gamelan project at NIME 2005. At ICMC 2007, Essl and Rohs (2007) discussed ShaMus, an application that makes music using mobile sensors, such as accelerometers and magnetometers. At SMC 2013, Ekeus *et al.* (2013) presented an application that allowed the user to create musical patterns with the help of artificial intelligence based on Markov chains. At the same conference, Baldan *et al.* (2013) discussed an application to help children develop rhythm and timing abilities. At SMC 2015, a mobile application for multimedia interactions using common inputs like sensors and buttons that processed video input in real time for interaction with low-level features was described (Krekovic *et al*, 2015) — something feasible nowadays due to the advances in mobile device technologies.

One of the first reports on MM was presented at NIME 2006 by Gaye *et al.* (2006), describing the current field, the emerging community, and many past events. An update of this report is found in a paper presented at NIME 2013, that evaluated 10 years of MM projects and papers presented at NIME (John, 2013). The publication of The Oxford Handbook of Mobile Music (in two volumes), which includes a bibliography related to the research field, can be considered as the most complete publication regarding the use of mobile technology and the social aspects of its acceptance (Gopinath and Stanyek, 2014a,b).

The appearance of MM news in the media is also worth mentioning. A New York Times article dated 5 December 2009, "*From Pocket to Stage, Music in the Key of iPhone*", discusses mobile phone orchestras from Stanford University and Michigan University and the John Hollenbeck Large Ensemble, a big band using smartphones as instruments (Miller and Helft, 2009). "*Gorillaz give away their new album made on an iPad*" is the title of a post in The Guardian (25 December 2010) describing an album composed using twenty mobile apps and instruments synthesized in the Apple device (Smith and Doward, 2010). Bjork's "*Biophillia*" album and mobile app appeared in a post on 13 June 2014 entitled, "*iPad art by singer Bjork becomes first-ever app in MoMA's permanent collection*". This album turned out to be another important landmark in the media while described as *part album, part interactive multimedia presentation* (Hughes, 2014).

Smartphones, the improved versions of the original cell phones, are now taking the place of computers. Mobile processor power increases at the same time as miniaturization reduces the size of many components, and batteries last longer than before even under this condition. They usually combine many diverse technologies into a single smartphone board: as of 2010, several simultaneous communication technologies exist, many processing cores, large storage capacity, and dozens of sensors[1]. It is interesting to note that by 2006, one could find smartphones with WLAN, 3G, Bluetooth, infrared, GPS and radio technologies integrated into a single board[2]. These technologies

[1] Regarding technological upgrades throughout the years, the reader may find that, in the past 10 years, processors got almost 10 times faster in each core, memory capacity got 25 times bigger, and communication technologies increased the number of network bands and their bandwidth (Appendix A).

[2] Appendix B shows sensors available for mobile devices, both base sensors (the ones related to physical sensors) and composite sensors (which build upon values from base sensors, or merge two or more sensors).

make this device a suitable option for collaborative and interactive ideas.

2.1 Background: Network Music

As soon as 19th century technologies became capable of reproducing and playing back audio, even before the digital age, communication technologies inspired musical and artistic experiences. Audio transmission and control between distant places through telephones and radios were just the beginning, before the introduction of computers. In this context, network music can be defined as music that takes advantage of network technologies to be composed, played, or simply exist. The term is common in academic publications and also used as computer network music, although the music based on using computer networks lacks universal terminology (Akkermann, 2014).

Computers have been used for network music since the 1970's. Switching computers to phones for concerts during the 1990's made network music a possible mother field of MM, or at least the most important predecessor. Smartphones appeared after cellphones and included technologies such as advanced operating systems and browsers for Internet access. Smartphones can thus be used the same way computers can be, including the composition and sharing of network music, but with the added convenience of mobility.

One of the first setting decisions for network music consists of determining whether the composition would work for local networks or through the Internet, or even both. Note that local networks are mostly associated with a local area network (LAN), but even this setup allows computers located far away from each other. Musical performances using LAN in the context of a dozen meter radius might take advantage of small packet loss and fast packet exchange. On the other hand, performances using the Internet or wide area networks (WAN) can have users from different networks. Additionally, network music experiences have also been using other technologies, like radio networks and telephone lines, to extend the possibilities of interaction, as we will see below.

John Cage was one of the first composers to realize the potential of using electronic technologies to improve musical experiences (Weinberg 2003, p. 16-17). Cage's "*Imaginary Landscape No. 4*" can be considered the first *electronic interdependent musical network* as it was composed for 12 radios played by 24 artists. In this composition, the radio network was the source of sound, and the performers were guided by a score to select specific radio frequencies, one at a time during the performance (Weinberg 2003, p. 24-25).

Musicians John Bischoff, Jim Horton, and Rich Gold created The League of Automatic Music Composers in 1977, taking advantage of the arrival of the personal computer, particularly the *Comodore KIM-1*. The League was the first group to incorporate this technology into musical performances. Each member was able to send and receive musical data through the network created between them (Weinberg 2003, p. 29). In the mid 80's, the League evolved into The Hub group, which had more precise communication schemes using central computers and the MIDI protocol to facilitate the interaction. They also tried remote collaboration through telephone lines during this time.

From 1979 to 1983, a series of festivals took place in the imaginary city of Wiencouver, located metaphorically between Vienna and Vancouver, with events simultaneously in both cities. In its fourth edition, the *Wiencouver IV* in 1983, the festival organizers presented the project *Telefon-Musik*, in which performances were transmitted through telephone lines before being amplified to the public (Corby, 2013, p. 19-20). *Wiencouver IV* is the last one in its series and, with the *Ars Electronica* festival, it represents events in which the technology could be explored for collaborative performances at that time also using slow-scan television and telefacsimile (Gere, 2006, p. 146-147).

Using an ATM-based LAN and SGI machines running IRIX, Gang *et al.* (1997) implemented and tested the prototype of a TransMIDI system so that users could join network sessions and share MIDI messages asynchronously. This solution applied Multicast groups in order to allow cooperative networking applications with many layouts and topologies (Gang *et al*, 1997). For another project, the researcher Sergi Jordà was invited to create the FMOL project in 1997 for collective composition. Since 1989, he has conceived virtual instruments and interactive systems and built a tool that would

suit musicians and hobbyists with a wide range of complexity and creativity. This system required a computer with any 16-bit multimedia sound card and a 16-MB Pentium 100 MHz with Windows 95 and the Microsoft DirectX libraries that allowed up to four composers for real-time synthesis, composition and control (Jordà, 1999).

In the next decade, the popularization of laptops inspired a new type of ensemble usually referred to as *laptop orchestras*. In these arrangements, it is possible to interact through music and chats, projections, code and file sharing over a network (Gasperini, 2011). Most of these orchestras were founded inside academic environments, such as Princeton University, Stanford University, Georgia Tech, Virginia Tech, and Louisiana State University. Other bands, such as *Benoît and the Mandelbrots*, perform on stage using only laptops at the site of the *International Association of Laptop Orchestras* (IALO, 2016).

Performances with laptops caught audiences' attention due to its innovative setup, but it also has a drawback: during one performance one critic insinuated that the orchestra's performers were checking their emails and it was affirmed that musicians and the music were disconnected regarding physical expression (Trueman, 2007).

Interaction with traditional musicians is another achievement of these orchestras and seen with laptops generating musical notation for musicians holding acoustic instruments and communicating through the network (Lee and Freeman, 2013). In fact, an extensive environment can be achieved when using computer networks for laptop interactions with traditional ensembles.

Network music experiences and technological advances opened up many possibilities for musicians and scientists. This technological evolution inspired researchers to evaluate and theorize about network music. The works cited below are important papers that present great discussions, paradigms, insights and numbers regarding network music in Brazil and around the world.

Although little research was available about internet music conception from musical and technological perspectives 20 years ago, the cutting edge technology and its rapid progress were setting the stage for internet music to soon become a reality or virtual reality (Kon and Iazzetta, 1998). Audio compression or synthesized music processes would need to improve to diminish delays inherent in interacting through the Internet and allow interactions between musicians and their audiences in real time. Expected solutions mentioned in the paper are now a reality, including technologies such as IPv6, gigabit Internet connections, and super computers. The authors conclude by suggesting three factors that need attention for internet music compositions: pulse, tempo, and style. They also emphasize that composers need to compensate for technological limitations throughout their own composition (Kon and Iazzetta, 1998).

Paradigms related to music and networks are discussed in terms of aspects like performance, convergence, invention, subjects, plurality, and many other characteristics intrinsic to network music (Tanzi, 2001). Dante Tanzi presents network music structures from different points of view, including that of the composer, performer, audience and listener, that can be situated at different times in different spaces from the performance perspective where presence on the Internet is much like real presence in some cases, and virtual presence in others, correlating this discussion with the previous ideas (Kon and Iazzetta, 1998).

Human perception has improved through centuries of sound experiences (Jourdain, 1997). Following this idea, network music deserves attention regarding its effects on audiences and participants. The effects of latency and jitter have been studied and surveyed for the past focusing mainly on perceived synchronization and they present parameters to bear in mind while proposing musical interactions. In these works, authors demonstrated that human perception is constrained by some limits of rhythm, haptics, and other limitations discovered and evaluated through many careful experiments, whereby latencies up to 30 ms are acceptable for multimedia applications, but higher latencies may affect interaction (Lago and Kon, 2004; Rottondi *et al*, 2016; Schuett, 2002).

Many works using the Internet for music interactions were also grouped and discussed in papers at the turn of the last century. Weinberg (2003) devised four categorizations: Server, Bridge, Shape, and Construction Kit, based on participants' interconnection levels and the role of the computer in this environment. Presented also as a basis for new ideas relating to network music, this research

introduced the concept of Interconnected Musical Networks (Weinberg 2005). In these networks, a musical instrument can be played by more than one person simultaneously and many musicians can interact with the music system through any network system. Although this concept was wide-ranging in its scope, it embraces the new network solutions as valuable elements for new musical instruments.

Alvaro Barbosa theorized about network music with a focus on network systems and collaboration and proposed a classification diagram based mainly on musical interaction and its location (Barbosa, 2003). The interaction methods are either synchronous or asynchronous, while the location can be co-located or remote and combining interaction methods and location, he found possible characteristics of collaborative systems from strict synchronized local performances to remote free improvisations. This mix also provided new sonic art creation paradigms toward network interaction.

The question of presence on the Internet and the possibilities of dealing with space and time constraints were technically scrutinized by following the technological advances of network communication, and analyzing structures for network performances (Carôt and Werner, 2008; Carôt, 2010; Carôt and Werner, 2007, 2009; Carôt *et al*, 2007). Models based on specific approaches were defined by Alexander Carôt, such as realistic interaction, master/slave, laid back, latency accepting, delayed feedback, and fake time, which are discussed and exemplified in Section 2.4. These models are somewhat similar to Gil Weinberg's and Alvaro Barbosa's models, but focus on audio latency for each participant. All of these papers provide clues to better understand network music ideas and limitations.

In addition to theoretical discussions, network music ideas have been frequently put into practice during the past decade. The diffusion of the megabit Internet benefited distributed music projects, while providing fast communication between distant places. The concept behind distributed music is that when you distribute content, there may appear to be similar music in every place, but the music is unnecessarily the same. By taking advantage of local networks or the Internet, researchers improved ideas presented in the beginning of this section, and these improvements occurred around the world and also in Brazil, as we will see below.

Laptop orchestras started to display collaborative remote live coding into their performances. Swift *et al.* (2014) used a network tunnel — with Secure Shell (SSH) — through a server in Australia to conduct a live coding performance that included performers situated in Germany and USA. Roberts and Kuchera-Morin (2012) and Ogborn (2014) have been using Internet browsers to share code among connected performers, with Gibber and Extempore, respectively.

Simultaneously in Brazil, Mobile and NuSom research projects from the Universidade de São Paulo (USP) produced distributed music concerts as well. NetConcerts (Arango, 2014) displayed musical collaboration between members of those projects from São Paulo, Brazil, and the Sonic Arts Research Centre (SARC), Queen's University Belfast, Northern Ireland. Researchers discovered many challenges specific to Brazilian reality in establishing a reliable communication at network music performances due to connectivity constraints (Arango *et al*, 2013; Arango, 2014). One of these researchers evaluated the tools used during these concerts and, comparing with available solutions, came up with the conception of JackTripMod as an alternative solution for home users aiming to perform using the Internet (Tomiyoshi, 2013).

In another project from USP researchers, the Medusa system was developed as a distributed music environment (Schiavoni and Queiroz, 2012; Schiavoni *et al*, 2011, 2013). This system permits users to share audio and MIDI data among connected computers using many network protocols and audio API's (Schiavoni, 2013). In addition, I created SuperCopair as a tool for collaborative (de Carvalho Junior *et al*, 2015a) and cooperative live-coding (de Carvalho Junior, 2015) through the Cloud that was developed in a personal partnership with members from the University of Michigan, Ann Arbor, USA. Through this tool users can share and run codes that synthesize music in a local or distributed manner. The focus is to make it easy for users to practice distributed music without many steps or previous technical knowledge.

Network music changed musical concert paradigms beginning in the 1970's. Many new technolo-

gies were used during this time to interconnect desktop computers, laptops, and supercomputers. Introducing smartphones into this field provides mobility, easy access to sensors and touchscreens, and also focuses attention on wireless connection limitations and battery consumption. The use of smartphones in network music and the development of MM are discussed in the next sections.

2.2 Devices overview

Mobile device systems are constantly being upgraded around the world. On one side, the market (e.g. Apple and Samsung) has a patent warfare resulting in discussions regarding who invented the new feature "first", while on the other side, users and researchers are updating their systems as soon as possible, while also creating open source alternatives.

Examples of MM open source alternatives for smartphones are the Android system, libpd library for Pure Data, Web Audio API, and accessories like Arduino connected through USB On-The-Go (OTG). Although only new devices are closely related to these new features, smartphones are becoming similar to desktop computers from a hardware plug-and-play perspective.

Nowadays, the evolution in mobile devices' processing and computing capabilities follow a similar pace as personal computers, even though their purpose and usefulness are considerably different. Mobile devices have been reduced in size, increased in power, save energy, and efficiently dissipate heat with the benefit of continuous research that also improved other computing systems such as servers and desktop computers (Barroso and Hölzle, 2007).

The mobile evolution also involves the possibility of combining other mobile hardware, such as cameras, communication interfaces, and sensors. These are features that appear from time to time, which, once available, attract the attention of performers and musicians due to their artistic affordances. The musical use of such technology allows the inclusion of both sensors and network data into musical processes that can include gestural control and interaction between users.

Hardware advances have spurred on software advances as well. The operating system (OS) is one of the most important pieces of software on mobile devices. Symbian OS and Palm OS were available on devices used in mobile music projects during the early 2000's, while iOS and Android established their places on the market afterwards. Initially, mobile devices running iOS had an advantage regarding audio latency compared to Android devices, due to Apple's development of both hardware and OS, but nowadays devices running an Android system have low latency in new versions of the OS, even with different hardware (Bianchi and Queiroz, 2012).

Following advances in OS, libraries, and languages, mobile devices have achieved high quality on audio synthesis, processing, and reproduction. The portability of computational solutions from desktop computers to mobile devices has accelerated this process. Most notable works include Pure Data application for Pocket PCS (Geiger, 2003); Pure Data patches compiled with J2ME (Schiemer and Havryliv, 2005); the development of libpd, a library to run Pure Data patches in many programming languages (C, C++, C#, Objective C, Java, Python) and systems (Android and iOS) (Brinkmann *et al*, 2011); Mobile STK (Essl and Rohs, 2006) ported from Cook and Scavone's STK (Cook and Scavone, 1999); the use of ChucK on the Ocarina app (Wang *et al*, 2008) for iOS; the port of CSound for Android devices (Yi and Lazzarini, 2012); and the iSuperColliderToolKit developed for iOS (Ito *et al*, 2015). These works are further described in the next section.

Android device programmers can use low level native code for audio processing without any library to improve performance, and this alternative performs better than available options for Java (Bianchi, 2014; de Carvalho Junior *et al*, 2013). Although loading native code takes 1 to 4ms, the native code surpasses pure Java in most situations. Moreover, the multi-thread version of native code has similar performance to the single-thread version, due to Android threading policies, rendering multi-threading unnecessary.

The use of Internet browsers for MM is another approach that requires attention and offers advantages for many mobile platform (Wyse and Subramanian, 2013). From web pages to web applications, researchers can create interactive projects that allow users to control audiovisual parameters during performances (Allison *et al*, 2013; Weitzner *et al*, 2012). Further development of audio APIs

currently allows browsers to process and synthesize audio inside webpages. During the 1990's, web pages only supported Wave and Au files, but now they can play lossy and lossless formats, such as MP3 and FLACC, respectively. Audio streaming has been an alternative since the beginning of the Internet. Duckworth (2005) provides a detailed description of most of the initial concepts and works related to the use of music for the web.

Other audio resources on browsers include audio synthesis through Web Audio API with high performance and flexibility, even on mobile browsers. Applications with Web Audio API are turning web pages into desktop software, like the interactive audio renderer (Matuszewski *et al*, 2016) and DAWs (Kleimola, 2015). Web Audio libraries like Gibber (Roberts and Kuchera-Morin, 2012) and WAAX (Choi and Berger, 2013) have facilitated the use of music technologies on browsers by reducing the complexity of coding required and the interface design. Moreover, the majority of new mobile devices can run Web Audio API applications through browsers, which makes it portable to many systems and platforms.

In terms of MM interaction, applications can apply direct or indirect audio manipulation. Apps using buttons, visual controls, and touchscreen permit direct manipulation as the interaction depends on user's actions, i.e., the user decides when to send new inputs or parameters to the application through the interface. The application "Touches on The Line", described in Appendix E.3, applies the idea of direct manipulation. The touch position acts as an input to the CSound synthesizer, and sound is generated following the position of the fingers on the screen, before being sent to other users who will reproduce the same audio (de Carvalho Junior, 2013).

Developers take advantage of sensors, gestures, and multimedia resources like audio and camera for indirect audio manipulation on mobile applications. Although these inputs sometimes require specific actions from the user's side to interact with MM, their current values are sent with a fixed (or variable) sample rate to the application. This situation constantly provides new parameters to the application, even if the user stops interacting with the mobile device. The application "Sensors2PD" described in Appendix E.4 is an example of indirect manipulation, where users can control which sensor sends values to a Pure Data patch, so sensor events are sent when they are acquired by the system (de Carvalho Junior, 2014).

It is also possible to have mixed interaction when applications allow direct and indirect manipulation of audio parameters. Ocarina (Wang, 2014) uses this in its approach whereby buttons manipulate the tone of the output, while, at the same time, the audio input is used to capture the blow from a user's mouth to control the amplitude of that tone's synthesis. The project "Hoketus" (Bandeira and de Carvalho Junior, 2014; de Carvalho Junior, 2015), described in Appendix E.5, uses available wireless network information to define the sound synthesizer, to modify synthesizer parameters through touch on a smartphone screen, and to allow the use of accelerometer values as parameters at the same time. The direct, indirect, and mixed manipulation definitions here are far from defining a general taxonomy, although they correspond to HCI concepts with similar descriptions.

Given all of these technological advances and features, we will discuss the use of mobile devices, especially smartphones, as mobile instruments in the next section. Problems faced by developers and musicians are presented with more detailed information regarding musical aspects. Projects and companies that encouraged the boom of MM in recent years are cited, as well as past distinguished researchers from the field.

2.3 Devices as Music Instruments

The insertion of mobile devices in the ambit of musical instruments has been an important approach for the digital musical instrument (DMI) concept (Miranda and Wanderley, 2006). The representation of a DMI runs from the input to the sound production, including a gestural controller, mapping section, primary feedback from the gestural controller, and secondary feedback from the sound production. This approach to represent a DMI is similar to the process of generating sound on acoustic instruments. However, fixed causality is optional, so the same input can be mapped

each time to a different sound, if so desired (Miranda and Wanderley, 2006). The possible ways of using smartphones discussed in Section 2.2 can place them in any position of this chain without the necessity of defining a complete DMI.

Adopting mobile devices as musical instruments faces some restrictions. During musical performances and musical practices, real-time signal processing and network data streaming require attention. Processing high quality audio on mobile devices in real-time while applying effects on many audio sources may result in undesirable glitches due to lack of CPU resources. Despite that, companies are constantly improving device hardware and software, and offering new options for composers and performers. The composers can also avoid burdensome strategies until it becomes feasible to running them on a specific device.

At this point, a recommended strategy to create musical instruments with mobile devices is to take advantage of computer music solutions imported from desktop computers, like Pure Data, STK, Chuck, CSound, and SuperCollider, and also the Web Audio API. The migration of code and creativity from these solutions to mobile applications is thus accomplished without much rework. Available computer music languages, libraries, and the Application Programming Interface (API) take care of mobile constraints, settings, mobile resources (like sensors), and dealing with audio input and output even when specific hardware for Digital Sound Processing (DSP) is unavailable. Additionally, research has also created toolkits and systems to facilitate the development of musical applications encapsulating language and library compatibilities in order to make the work easier for novices of MM technology. As the tendency of using mobile devices as musical instruments increased, there were many MM projects using the solutions discussed here, which are presented below.

Miller Smith Puckette's "Pure Data" language (Puckette, 1997), also known as Pd, has been integrated with many MM projects since the beginning of the 2000's. The first integration was Günter Geiger's port of Pd to Pocket PCs running Linux (Geiger, 2003); using the PDa application users would open and interact with patches through the touch screen. The author also discusses the situation of mobile technologies at the time of the paper asserting that "having a new technology available to generate music is always a challenging experience, only the future will show what kind of application this will have" (Geiger, 2003, p. 4).

Schiemer and Havryliv's "Pocket Gamelan" is another project that used Pd on mobile devices, allowing users to compile patches for the Java 2 Platform Micro Edition (J2ME) — a Java version available in the very first smartphones. Taking advantage of pd2j2me application, composers could adapt Pd patches to mobile devices without writing any Java code. This project also created performances using Bluetooth in order to interact with other devices running the same application.

The development of libpd was an important advance in recent years. This library allows loading of Pd patches inside programs written in Java, Objective C, and other languages. This solution encouraged the use of Pd on Android and iOS applications. Examples of these applications are, for example, RjDj and PdDroidParty (Brinkmann, 2012; Brinkmann *et al*, 2011). During our research we also created apps using libpd, like the *thereminimal*, *Hoketus*, and *Sensors2Pd*, described in Appendices E.1, E.5, and E.4, respectively.

"WebPd" from Piquemal and McCormick is a recent project that allows the use of Pd patches on diverse systems through a browser (Piquemal and McCormick, 2017). It is a JavaScript library that interprets some Pd elements from the patch, projects them on web pages, and runs the patch using the Web Audio API. This project is an alternative for those who want to perform using web browsers and Pd without dealing directly with Web Audio API.

Based on the "Synthesis Toolkit", also known as STK (Cook and Scavone, 1999), Essl and Rohs developed the "Mobile STK" (Essl and Rohs, 2006), which was first used on devices running Symbian OS during the development of the CaMus application (Rohs *et al*, 2006). The application was used at interactive musical performances in which visual markers could be detected from the device's camera, allowing the control of musical parameters while sending MIDI data through Bluetooth. CaMus 2 included the interaction between the participants during the performance and on-screen information over the camera image (Rohs and Essl, 2007).

The Mobile STK was also used on ShaMus (Essl and Rohs, 2007). The main difference here is the interaction using new sensors available on mobile devices. The accelerometer and magnetometer were somewhat imprecise back then, so researchers attached an external device to the smartphone for accuracy regarding the mobile device position. In subsequent research, the audio input was added to the system allowing onset detection and interaction based on audio amplitude (Ananya *et al*, 2008). The STK was also used as part of the MoMu toolkit and the urMus mobile music environment.

The Mobile Music Toolkit was the first port of the STK to iOS, motivated by the conception of the iPhone 3G by Apple and the advantages of using the iOS SDK over the Symbian SDK, previously used by the authors for their applications (Bryan *et al*, 2010). Their justification was that the former SDK provided tools for agile project development, while the latter was hard to use for new developers. Although the MoMu was heavily used by students from Stanford University during the time of its conception, as described by the authors, its last update appeared in 2010.

In turn, the urMus is a mobile application that can be used to create other mobile music applications using Android or iOS devices and a browser. Users can program interactions and interfaces using the Lua language for their applications, and accessing the mobile device from the browser, before loading the code into urMus. Network interaction is also allowed through Bonjour or TCP connections. The system permits the use of all available sensors and input/output interfaces like audio, camera, and GPS. Since its development, urMus has been used in many research projects as like a learning environment for percussive music collaborations (Derbinsky and Essl, 2012), and audience participation using the Mobile Ad-hoc Network (MANET) (Lee *et al*, 2014). Many urMus applications were also created by students as their final project of the "Building a Mobile Phone Ensemble" course, from the University of Michigan, since 2009 under the direction of Prof. Georg Essl, which resulted in some performances, the last one being in 2015 (CSE News, 2017).

One of the most prominent MM applications is the Ocarina for iPhone (Wang *et al*, 2008). The ChucK language (Wang *et al*, 2003) is embedded in this application to provide audio processing functionalities. User interaction is made through the blowing on the smartphone's mic while the musical notes are defined by four circles on the screen, simulating holes that can be closed by touching the corresponding circle, as in a physical ocarina. The application was on the Hall of Fame of Apple Store for many years, and today users can still play the legendary ocarina on iPhone devices and listen to other users playing around the world (Wang 2014).

During the 2010's, mobile devices became also compatible with one of the oldest computer music languages, CSound (Vercoe and Ellis, 1990). Available for Android and iOS devices, many applications have been developed using the mobile API and are distributed as demonstrations inside the CSound mobile application. During our research process, we also used the CSound mobile API to create an application named "Touches on the Line" in which user's touch movements are shared through a webserver to all connected users, so that they can listen and play together. The audio synthesis is made with pure CSound, while the server uses Ruby on Rails (RoR). The application is described in Appendix E.3.

CSound files can also be loaded on web browsers through the use of the Emscripten compiler, that compiles C/C++ into JavaScript applications compatible with Web Audio API, and also through the use of the Portable NativeClient (PNaCl), that compiles C/C++ into abstract architecture-independent executables. This web interface is compatible only with specific browsers and is described in Lazzarini *et al.* (2014).

Mobile applications using the language "SuperCollider" (McCartney, 2002) have recently been conceived and reported in some research projects. The AuRal system is an example of an Android application that allows users to create ad-hoc ensembles based on geolocation. The SuperCollider server running on the mobile application receives OSC messages with parameters used during real-time music generation (Allison and Dell, 2012). The iSuperColliderKit is an alternative for iOS programmers. This project provides an embedded version of the SuperCollider server and interfaces for communication through the Objective C and Swift languages. SuperCollider code fragments can be sent to the server through native language methods and are interpreted and synthesized afterwards (Ito *et al*, 2015).

The idea of using mobile phones as musical instruments goes far beyond simply creating applications to be used during performances. The first mobile phone orchestra was conceived in 2007 at Stanford University and is known as MoPhO (Wang *et al*, 2008). In the beginning, they used the Nokia N95, a revolutionary smartphone at the time in terms of functionalities, available buttons, and number of sensors. Later they changed to iPhone devices due to iOS SDK advantages already explained in this section. The musicians amplified the sound of smartphones with speakers attached to the arms and plugged to the mobile device. The interaction involved moving the device around, touching its screen, pressing the buttons, or using resources like microphones, speaker feedback, camera, sensors, and also communication technologies like Bluetooth and WiFi. Siblings of the MoPhO were then created by its idealizers. Georg Essl conducted the Michigan Mobile Phone Orchestra at the University of Michigan, while the Helsinki Mobile Phone Orchestra was directed by Henri Penttinen at the Helsinki University of Technology in Finland. This movement inspired the conception of new orchestras at other US universities and countries like South Korea and Japan.

A reference application that was used to propose a collaborative performance with audiences as musicians was the "echobo" (Lee and Freeman, 2013). This mobile instrument had an audio engine based on the MoMu Toolkit, the user interface created on COCOS 2D API, and the server programed in PHP language. The user would interact during a performance through a piano-like interface using the touchscreen of an iOS device. A musician guided the harmony of the performance by sending chord information to all devices connected to the same room on the application. One of the performances included more than 100 participants and also a traditional musician playing a clarinet on stage.

During our research, in partnership with Sang Won Lee and Georg Essl, we conceived another application for audience participation named "Crowd in C[loud]". In this case we took advantage of Web Audio API to create an application that could be accessed from web browsers, avoiding any installation process and presenting compatibility with most mobile's devices OSs. A musician would also guide the audience harmony from the stage, but, differently from echobo, the devices were interconnected using Cloud Services. Doing this, we allowed users anywhere to join the performance without any network setup besides entering a web page URL. The scalability of Cloud Services is another advantage of this approach that permits a high number of users to interact, while the distributed Cloud clusters shorten the interconnection paths depending on each user's location. More details regarding this project are presented in Appendix E.8.

These selected applications represent particular cases of MM technologies. The fact that some of these applications are aimed at interactions between users implied the use of network technologies in most of the situations to allow data exchange between devices. In the literature, there are also works with mobile controllers interacting with central sound systems through local network or Internet (Allison *et al*, 2013; Hindle, 2013; Weitzner *et al*, 2012). Considering the applications cited in this section, we have communications interaction and collaboration. Although the communication structure is preceded by decisions regarding technologies, settings, and setups, these principles are better related to an artistic strategy. These three keys together deserve some attention regarding MM instruments.

2.4 Communication: Interaction and Collaboration

The term 'mobile' in MM is related to the idea of being able to move. In this case, the music of the musician is moving from one place to another during the music timeline. Movement is inherent to communication technologies. This field improved interaction paradigms by permitting user's intercommunication with network servers, computer machines, and distant users.

Users may have different constraints in MM interactions depending on the channel and noise during communication. Data exchange constraints can also affect the idea of synchronization in collaborative MM experiences. Papers discussed in Section 2.1 determined that network interaction suffers with delays higher than 30 ms (Lago and Kon, 2004). One example is a call using optical fiber between Brazilian extremes, North-South or East-West, through a path of approximately 4,000 km.

The delay in this case would be 20 ms, which is almost acceptable.

As previously cited, Carot's approaches are described as realistic interaction, master/slave, laid back, latency accepting, delayed feedback, and fake time, and all of them are related to the way users manage network constraints and interact throughout a performance. These approaches can also be applied to MM projects that focus on collaboration through any network. The Realistic Interaction approach considers musicians sharing the same physical space. This model is dubbed as realistic because the data transmission is similar to the sound diffusion during a real performance on a stage. SWARMED and NEXUS projects are examples of MM following this model. In SWARMED (Hindle, 2013), Abram Hindle used a captive portal to capture participants' attention as they had restricted Internet access through the portal and were concentrated on the performance. The NEXUS project focused on distributing unique interfaces to users accessing a webpage on their mobile devices (Allison *et al*, 2013). In this case, events from user interactions were exchanged by a central server based on Ruby on Rails (RoR). The server was communicating using OSC with a synthesizer made in the MAX/MSP application.

The Master/Slave approach is applied when there is high latency between participants of an MM performance. In this case, one of the participants acts as a master, executing their part during the performance without waiting for the slave responses in real time. CloudOrch, TweetDreams, and Crowd in C[loud] are some examples of this approach as the interaction occurs through external servers that provide latencies higher than 25 ms in practice. The CloudOrch was proposed by Abram Hindle and deployed virtual machines for client and server users. In this project, the audio was streamed from cloud instruments to both desktop computers and mobile devices. While the cloud server acts as a master sound card, the devices act as slaves using web browsers and websockets for intercommunication (Hindle, 2014). At TweetDreams, the user contributes to the main performance through the Tweeter system by sending specified search terms translated into musical and visual reactions (Dahl *et al*, 2011). Python, Chuck, and Processing languages are used in this project to retrieve tweets, generate melodies, and render the graphical interface, respectively. The Python application acts as the master application, while the slave is the Tweeter application. Crowd in C[loud] (de Carvalho Junior *et al*, 2016; Lee *et al*, 2016), described in Appendix E.8, is a performance where the master stays on stage while the slaves are the audience's mobile devices acting as performers.

In the Delayed Feedback approach Carot defines that, independently of delay, all inputs are in sync and the participants think they are working together. massMobile is an example of this approach as authors take advantage of time-stamped messages in order to minimize the effects of latency, playing all events together based on the time-stamps. This project also takes advantage of a robust cloud based Java server and has been used in several performances as a client-server alternative for audience participation while accommodating the unpredictable latency common in MM (Weitzner *et al*, 2012).

In the Fake Time approach, the issue with latency is avoided. The musical events are only played at the next musical bar and the participants play with events during the previous bar by other participants. The project 'Touches On The Line' described in Appendix E.3 can be considered an example of this approach as finger movements on the touchscreen are sent only when the finger releases the screen (de Carvalho Junior, 2013). Although the application was conceived for distributed performances, when all devices share the same physical space, musicians can hear what is played and then react consciously.

The Latency Accepting approach is applied in cases when the composer and musicians do accept latency and packet loss as part of the performance. This model of collaboration is more relaxed regarding network constraints. The application Sensors2OSC described in Appendix E.4 is an example of this approach (de Carvalho Junior and Mayer, 2015). The data is exchanged within this application using the UDP protocol, which lacks packet delivery confirmation and may display high latency depending on the network.

Considering short range intercommunication for MM, we can take specific technologies into account, such as infrared, Bluetooth, and WiFi. The first is rarely available on mobile devices but is

one of the cheapest interfaces for communication. Bluetooth technology allows wireless communication with close devices in a range from 10 to 100 square meters and the devices are free to move during the connection. WiFi is the commonest of these three and the communication range can exceed hundreds of meters with local connections available in open places, which is sufficient for short range situations. In this case, the compatibility depends on the WiFi standards available at the router. Although current devices can reach gigabits per second of data transfer using 802.11 ac, megabits per second solutions are more common and compatible nowadays.

Wifi and 4G are good solutions when long range communication is required. The main advantage of 4G devices is their backward compatibility with 3G, and both 2G technologies EDGE and GPRS. On the other hand, the 4G signal can be reduced or unavailable in indoor situations depending on the antenna's location. Considering the antennas, the ability of keeping the connection when in motion is achieved by switching between WiFi antennas (through the use of repeaters) and with a handover process in 4G.

Internet access may be required for short or long range communications during collaborative practices. Infrared, Bluetooth, and WiFi allow access to the Internet through the use of other devices such as switches or access points, while the 4G technology offers Internet from the service provider. A common approach is the use of servers in order to interconnect the devices, and the advances in Cloud Computing encourage its use. Cloud solutions can provide features like scalability and also different levels of services such as infrastructure, platform, software, monitoring, and communication.

From the perspective of internet music, once connected to the Internet, a device can connect to an unlimited number of devices. Musicians and users of these technologies are mostly unaware of their technical details and constraints. With this in mind, a more detailed discussion regarding the technical settings of MM networking technologies is provided in the next chapter.

Chapter 3

Mobile Networks

Around the world,
Around the world,
Around the world,
Around the world.

Daft Punk

The communication concept reached the current millennium with many small devices communicating through wireless broadband. On the other hand, the communication medium used may physically define the throughput based on routes and the amount of information for exchanged data. In terms of computer networks, although it is possible to use a wired connection, mobile devices are mostly free from cables to favor its own mobility aspect and they require batteries to maintain this condition. These two points are important for mobile networks due the restrictions they impose: wireless connections are subjected to noise more than wired ones; and power-saving strategies allow longer data transmission avoiding communication breakdown by full discharging. Some details regarding computer networks will be discussed in the following paragraphs and sections considering mobile devices communication and connection with other device types.

While WiFi networks depend on wireless routers that cover tens of meters with Internet access, the mobile networks cover tens of kilometers from the base station. As the culture of being "always on" is getting more prevalent, range and quality of mobile networks need to be increased constantly due the user's data traffic necessity. When the mobile network is unstable or presents low quality, switching from mobile networks to WiFi networks is thus the option, even if the user needs to stop physically somewhere just to get online, losing their mobility in exchange for Internet access and that "always on" feeling. However, it is important to notice that WiFi networks normally present higher throughput than mobile networks and allow for use at places in which mobile networks may be unsupported, such as inside concrete buildings that would prevent the direct passage of a magnetic field from a mobile network antenna.

In order to take advantage of WiFi or mobile network connections, a device needs a specific network interface to allow communication through a physical medium and is part of the physical layer of any network. Mobile devices intercommunicate through electromagnetic fields over the air or space with their base stations (or routers), while the wired connection from the base station is made through any other medium, such as fiber optics. Infra Red, Bluetooth, and USB interfaces that can also work as physical interfaces for mobile devices as these interfaces can be used to connect to a router device connected to a network, e.g., a computer sharing its Internet through the USB port with a phone. Depending on the interface, different protocols are used to exchange the bits between the interfaces, but the link layer is responsible for deciding how to pack the data depending on the interface available (and connected to a network) when the network layer wants to send this data without knowing the physical layer. When receiving data from a physical layer, the link layer verifies if the data needs to be resent through the physical layer or routed to the network layer.

Packets with data or partial data are moved from one network layer to another following rules

that depend on the destination address and routing algorithms. Many services available on the network layer are used to transport the packets in order to deliver the following rules defined from the transport layer. Guaranteed delivery is one of the services available at the scope of the network layer, and the IPv4 or IPv6 address works as a guide to define the route taken by a packet forwarded inside a network. Conceived during the 1990's, the IPv6 can help to correlate a single network address to a single device, as it used to be during the onset of IPv4 implementation, and then facilitate the transport of packets.

In the ambit of the transport layer, solutions for exchanging data on mobile networks may have to opt for solutions based on unreliable connections. The transmission rate varies while a device is moving depending on the signal quality, the distance from antennas, and communication interferences. Although unreliable connections inspire unreliable transport protocols such as UDP, the TCP can also be used, but requires a persistent connection in order to avoid reconnection for every data exchange. The main idea inside this layer is that there is a logical interconnection between two hosts where one or both can be using a mobile network, but none of them needs to know their actual interconnection path.

Mobile network applications assume the device will often change its IPv4 address, disconnect for a long period, and can discharge while connected. These circumstances require sessions to keep devices logged independently of a network address, data synchronization based on online state, and synchronization methods that download only important data from time to time or consider downloading heavy data online while connected through WiFi in order to save money on a mobile network data plan. In this case, dealing with unstable connections is a situation that requires good communication strategies from the application layer point of view. The protocols and interfaces used may thus interfere with the communication depending on the purpose of data exchange. If every single data is important, the devices may require a confirmation for each message sent, for example. On the other hand, the devices can just send as many messages as possible using protocols that are fast and unreliable if data loss is expected, planned, or desired.

Network application programmers and users need to adhere to conditions regarding data transmission between devices, as packet delay and packet loss are inherent to this context. In terms of communication, when a person decides to send a message, this person needs to decide the message encoding, the transmitter, the channel, and expects that the destination will share the same channel, use a receiver, and decode the message correctly. This concept is based on Shannon's Information Theory (Shannon, 1948), that is widely used on communication studies (Fiske, 2010). As the receiver may need to answer a message, probably the same channel and encoding will be used for communication to occur. Shannon also studied this scenario (Shannon *et al*, 1961), and the networks follow some of their results even today. Encoding a message effectively instead of sending the complete raw information has its effect as the channel will need to move less data. On the other hand, the channel selected may delay the message delivery or impose noise risk, so that the message may be delivered when it is unnecessary or even undelivered. The Information Theory was an important key for Internet development, and selecting the best network options for data exchange based on this theory is a desirable expectation for people who deal with network communication, or even any kind of communication approach.

This chapter discusses mobile network structure from devices to services. The devices used for communication on mobile networks are presented with a focus on mobility, power consumption, network constraints, and data exchange. Interfaces are presented with technical information for users intending to apply them, while the topologies formed using these interfaces are shown in correlation with classic computer music approaches. The protocols and services used in this thesis are also discussed in terms of their advantages and disadvantages, including specific constraints of some services chosen for this work specifically.

3.1 Mobile Devices

Computing data everywhere while moving is just one of the advantages of mobile devices. In the past, vehicles were used for transporting computers powered by batteries or power generators. However, the mobility got attention with the size reduction of components, that can reach nanometers nowadays in mobile devices This characteristic improved the capacity of mobile devices allowing more components to fit in a handheld board, with diverse communication interfaces, gigabytes of data, many sensors, and a long life battery. Although the devices can use all of these technologies in order to exchange data, the users of these devices face many issues related to battery and power consumption, data plan costs, handover, and data sync.

Choosing the smartphone to buy can be life-changing experience as the user needs to select brand, the features, the mobile network, the sensors, and all the other features that are available on the market options. Additionally, the impossibility of using many features at the same time due to power consumption have been caught the attention to batteries technology improvements as well. Power banks were developed in 2001, but they became a must-have accessory for smartphones only years later. The batteries are also improving, but as the smartphones are taking place of computers and notebooks — devices that are utilized with a power supply most of the time —, the smartphones are expected to work like them in terms of availability, requiring better batteries and recharging technologies.

Internet access is another requirement for most of users of smartphones. WiFi and mobile networks are available at multiple places and while the former is faster, the latter works as an alternative for situations where the first is unavailable or there is need of moving from one place to another, while connected to the Internet. A qualitative comparison between these two technologies is presented in Lehr and McKnight (2003). The authors assert WiFi as an alternative to mobile networks considering the cost for data transmission and consumption using mobile network. Even if the cost integrates the business model of the mobile network service providers, it affects both consumers and companies. The higher cost for installing a new antenna also led companies to invest in public WiFi hotspots for consumers that have an unreliable connection at some places, such as airports or shopping malls. In the same way, a performance comparison between the two technologies presented in Gass and Diot (2010) affirms that WiFi satisfies expectations in terms of download and upload while the long range in mobile networks surpass the WiFi settings.

Smartphones are expected to have battery charged and Internet connection access all the time. Users need to be synced with online data as fast as possible and data syncing requires real-time communication between the mobile device and many data servers. Instead of checking the servers all the time, the devices may apply some solutions such as a keep-alive socket for receiving notifications regarding any new data. Being connected all the time independently of the network interface or geographic location may result in switching between mobile network antennas or WiFi routers, or even between both mobile networks and WiFi technologies, additionally to recharging the device whenever possible. Furthermore, dealing with these constraints of mobile devices is inherent in daily life of smartphone users.

3.2 Network Interfaces

The way devices exchange messages has evolved throughout the time with diverse technologies being attached to the same device. Communication interfaces are compatible with many different devices and the communication quality does depend on the medium condition and noise interference level. From calling to high definition video conference, and from bits per second (bps) over infrared to gigabits per second (gbps) through WiFi, the mobile device users need to choose an interface to do some of these activities then so they can use WiFi to call someone or try Infrared to transmit a video conference.

Network interfaces differ in technical settings including transmission range and bandwidth. Short range interfaces can be used to predispose interaction between devices close to each other

and some examples are Infrared, Bluetooth, Near Field Communication (NFC), and WiFi. In case the user wants to communicate with a device far away, there are long range interfaces for this such as WiFi and 4G (or other mobile network interface such as 3G or 2G), however short range interfaces can also be used with the addition of signal repeaters or router devices. Following the Am, Fm, and television transmission methods, new wireless technologies such as mobile networks and WiFi routers connected to the Internet are examples of solutions for long range data transmission among mobile devices. Data throughput and bandwidth varies according to interface communication standard with ranges varying from bits to gigabits per second independently on the short or long range transmission characteristic.

Short range technologies

Short range communication has the advantage of face to face interaction and avoid distance constraints related to any communication through a physical medium. Data transmission and infrastructure are subjected to less issues regarding security as the devices will be close and the communication environment can be better monitored when all participants share the same place. The communication types can defined as Mobile-to-mobile, Fixed-to-mobile, Infrastructure-to-mobile, and Infrastructure-to-fixed depending on the devices participating (Deicke *et al*, 2012).

A common example of short range interface is the Infrared. Even a child can manage a remote control to change a channel on a television, while the communication is normally done by Infrared technology. The data transmission had no standard before the first meeting of the Infrared Data Association (IrDA) in 1993, when the first specification defined the communication as serial, half-duplex, and asynchronous in a range up to 1 meter with transmitter angle from 15 to 30 degrees, and the receiver angle up to 15 degrees. The bandwidth would vary from 2,400 bps to 115,200 bps at the first specification, and nowadays the GigaIR reaches 1 Gbps and in several meters.

Infrared interface has the advantage of being cheaper and efficient. The waves are invisible to human eyes and require low power for data transmission while the error rate as low as 10^{-9}. Infrared technology efforts are expecting new models that can be scaled up to 5 Gpbs and a new standard with 10 Gbps for broadcasting services (Deicke *et al*, 2012). One disadvantage of Infrared communication is that the transmitter and the receiver need to be facing each other while they communicate. This disadvantage also imply a single characteristic to Infrared due to its connectionless communication protocol: every device in the range of the transmitter will receive the data allowing a remote control to communicate at the same time with both a TV and DVD close to each other, for example.

Another wireless technology for short range communication is the Bluetooth (Bhagwat, 2001). This license-free technology was conceived as a single-chip radio operating in 2.4 GHz ISM (industrial, scientific, and medical) radio frequency band. The first specification was released in 2001 with throughput range from 36.3 to 585.6 kbps, although the link speed is 1 Mbps. The maximum distance supported for this specification is 100 meters though the higher the distance the higher is the power consumption (Gupta, 2013, p. 20), and the recommended distance is 10 m in the end. Current specification provides possible range up to 400 meters and throughput up to 2 Mbps over Bluetooth link itself (Bluetooth, 2016).

While Infrared requires line of sight, Bluetooth can communicate over barriers due to the radio frequency property. It is possible to have many devices interconnected through a master device or diverse networks interconnected through the slaves, and, additionally, current specification improves broadcasting and low energy communication. Bluetooth 3.0 High Speed can also exchange data up to 24 Mbps using Bluetooth link to establish the connection and finally communicating through a secondary radio available on devices, such as 5 GHz radio used for WiFi (Bluetooth, 2009). Although, the Bluetooth Low Energy approach is getting accepted as reliable for many solutions that requires to transfer just few data from time to time, the relation between power consumption and data exchange is still the main drawback of Bluetooth technology.

The NFC shows up as another option for mobile device communication with an one to one network topology and fast setup time (Coskun *et al*, 2013). Developed in 2002, the technology aims fast contactless communication in a range of 4 to 10 cm with a data rate up to 0.4 Mbps. Although

it is possible to have two mobile devices communication through NFC, in most of the situations the data exchanging is made from an active device that reads a passive NFC Tag and do "something" with the data read afterwards, such as open a website or confirm a payment.

The WiFi technology has many options for short range communication. Devices can interconnect through a home router and exchange data using the Local Area Network (LAN) using wired or wireless connection. In case of wired connection, the Ethernet network will depend on the available technology on all devices that currently varies from 10 to 1000 Mbps.

The wireless technology have specific details that depend on the antenna, router, and network structure. WiFi antennas have its power defined in decibels-isotropic (dBi) and the the transmission can be directional or omni-directional. The router needs to be compatible to give sufficient power to the antenna in order to use the full gain available. Additionally, the router can support different transmission standards, such as IEEE 802.11 a, b, g, n, ac, and many others. It is necessary to notice that two devices need to share the same standard in order to communicate, and this cause the industry to sell wireless routers compatible with as many standards as possible. The network structure can have router, switch, hub, repeater, and many other devices aimed to interconnect hosts. In terms of structure, the possibilities are infinite as long as one network can have its particular structure connected to other network defined in another particular structure.

WiFi Direct (Alliance, 2016) is an optional use of WiFi for short range interconnection. As the WiFi direct is mostly used by smartphones, the power of the antennas can offer a maximum range of 200 m and the network interfaces currently offer 250 Mbps maximum throughput, even if some new devices can have gigabit WiFi interfaces (Feng *et al*, 2014). The WiFi direct technology works like the hotspot, which is an option available in most devices that creates a network using the smartphone as a router. WiFi Direct is a peer to peer (P2P) connection and allows two devices to exchange data like in other short range solutions but using WiFi technology.

While there are many new short range technologies that works with dozen of meters range, they are designed for interconnecting devices close to each other. A short range technology can work for long range communication if one of the interfaces connects with another interface that would connect to a third interface and so on. In this case, the common setup is to have a device connected through a short range technology and another device connected to the Internet. This situation keeps the users close to a short range interface but communicating with devices far away, what may be undesirable in some cases, and it is similar to the use of telephones instead of mobile phones.

Long range technologies

Wired connections would be most suitable in some situations where the devices do have to stay in a small space or fixed place. The quantity of devices connected through wired connection is also another problem as the cable needs space. On the other hand, wireless connections would offer the advantage of being free of cables and allow users to keep moving freely while communicating.

Long range communication require specific network structure or interface: a short range interface can be used to communicate with an interface connected to the Internet or long range interface can communicate with an antenna far away. Although short range interfaces can be used for long range communication, they can consume more power in this unexpected situation. The most common long range interfaces are WiFi and interfaces used for mobile telecommunication such as 3G or 4G. These interfaces provide fast data exchanging and allow communication with antennas far away geographically.

WiFi technology works well as short and long range, and although the power consumption is almost 2 times higher than some short range alternative such as Bluetooth (Friedman *et al*, 2013), the advantage is the throughput that can reaches gigabits per second in some new standards and the range that can reaches dozen of kilometers with specific antennas (Raman and Chebrolu, 2007). Short range technologies are also restrictive due to the number of connected devices, while long range technologies can have a hundred devices connected to the same antenna at the same time. A wireless router for WiFi connection can have limitations regarding the number of connected devices even if most of them are able to offer more then 200 IP addresses. These limitations depend on the

routing algorithm and the antenna power that are technical settings related to device's model and brand.

The IEEE 802.11 standards define the possible WiFi technologies that can be available on any device, such as 802.11 a, b, g, n, and ac standards These standards are related to the network physical layer and they are available at IEEE Standards Association (IEEE-SA) website[1]. A WiFi router is able to implement different standards at the same time in order to be compatible with as many devices as possible, but the router needs to offer a high power to its antenna in order to have a long range signal strength and allow devices to move far away.

Another network interface available in mobile devices is conceived for mobile telecommunications. 2G to 5G are current options for mobile telecommunication. The name of the technologies are related to their generation, represented by the number in front of the 'G' which stands for generation. A new generation of mobile telecommunication technology is defined when a new standard is defined without back compatibility. A device compatible with 4G is able to connect to the older generations only when the manufacture implements the compatibility so that the device can switch between all of them, however the network interface of this device will dismiss any information regarding a new generation such as 5G. 2G has a power consumption higher than 3G and 3G consumes more power than WiFi, though it is expected that a new generation will always offer higher throughput and less power consumption than older ones (Balasubramanian *et al*, 2009; Rice and Hay, 2010).

The main advantage of these mobile telecommunication technologies are the range and mobility. Tall and powerful antennas are spread throughout the cities providing signal in every open space, on the other hand the devices can switch between these antennas transparently to the user. This characteristic provides more mobility for this technology without the burden of establishing connection or depending on extra settings or devices.

While connected to the Internet, the devices keep trying to switch to the best network interface for exchanging data in the area and saving costs. The costs here are related both to money expenses on data plan and energy consumed while communicating. It is expected that devices will switch between mobile data to WiFi when possible, and from WiFi to any wired connection as well. The network topology is also important at this point, as the device may be unaware of which technology has internet access and in this case particular setups are required for specific purposes.

3.3 Drawbacks in long distance networking

Network latency in the communication between devices from different locations offers a great barrier for those aiming music interaction between performers connected via long distance networks through mobile devices. Another barrier is the setup necessary to permit the interconnection between these devices in these networks through a wireless connection. In terms of multimedia applications, the latency need to be inferior to 150ms in order to looks like a synchronous interaction (Coulouris *et al*, 2011). A latency of 100ms and above implies some difficulty for network interaction even with experienced musicians (Bartlette *et al*, 2006), and if we have a sensitive ensemble performance, this threshold may be as low as 20ms (Chafe and Gurevich, 2004). Additionally, Lago and Kon (2004) says that 50ms is acceptable for chamber music, but the range between 20ms to 30ms is probable the perfect latency for remote musical applications.

The latency has a lower limit based on some constraints. If we consider the speed of light on optical fiber (200,000km/s) as a reference for data transmission and Earth circumference around Equator (40,000km) as the longest distance, a packet would take 200ms to finish the round trip. This result implies a latency of 100ms and it shows that interactions from different countries will have problems. One way to tackle this problem in local networks is using lightweight transport protocols such as UDP (Cáceres and Chafe, 2010; Harker *et al*, 2008), that has a low overhead when compared with TCP and fast data diffusion because it bypasses the establishment of connections and acknowledgment to each packet. In case we need to communicate with many devices at the

[1]IEEE Standards Association website: http://standards.ieee.org/ (visited on May, 2017)

same time, UDP has another advantage: it can be used in Multicast communication that we are going to discuss in Section 3.4.

The disadvantage of UDP is the possibility of packet loss during transmission. Applications that rely on UDP communication need some alternatives to take on this condition. The order of the packets is unpredictable in this communication as well and in some cases a missing packet is just a delayed packet that may can be discarded or not. These problems are some characteristics of UDP communication and they will probably occur more frequently on long distance connections than in a small local network. However, the setup of UDP is nearly network independent in case of a local or big network. A big network with the same services available in a local network is a suitable environment for applying solutions that were previously proposed for local networks respecting the distance constraints and the distribution of the devices to be configured when necessary. Academic institutions are connected most of the time through a research network that has direct connection between members and offers access to the Internet as well. The research networks from different countries are usually interconnected and they have support for network experiments when requested. The bandwidth of these networks are normally higher than popular Internet connections due to its research and experimental purpose.

An advantage of these networks for long distance communication is that their structure is well defined and they can be used as local networks with public IPv4 addresses and many services that are only available in local networks, such as the Multicast discussed in the next section. However, the setup necessary to start using these networks can difficult their use. In local networks we have normally one router and devices that are easily accessible in a physical and technical manner. Additionally, Academic networks are managed by many different partners and each point has its own rules and definitions.

Services settings depends on network managers. The routers have specific restrictions for security reasons, as most the routers are responsible for a huge data traffic from many universities and they need to avoid network attacks. The communication between the managers during a network setup is essential in order to minimize the risk of problems regarding a wrong configuration at some point. In the next section we will discuss the Multicast focusing on the restrictions of its use and setup inside academic networks.

3.4 Multicast

We have different network methods for exchanging messages using UDP. The Unicast is used to send messages to one and only one point on the network, while the Broadcast can be used to diffuse messages to all connected devices inside a network. Multicast permits packet diffusion to all devices that had subscribed to a multicasting group.

Multicast is available through various mechanisms, e.g. IP Multicast (Diot *et al*, 2000). Additionally, the network need to understand a protocol like Multicast Source Discovery Protocol (MSDP) in order to permit devices to find Multicast groups inside the network and create routes between them through the Rendezvous points (RP). The RPs defined on a network are the routers that serve as an encounter point from the information regarding group announcements and device's join requests.

Routers available in most of the houses are configured with Multicast natively and the users don't need to change anything. On the other hand, the same is untrue at academic networks. The routers available at most institutions on academic networks have specific hardware and software configuration due to the high number of computers connected through them. It implies many security settings and filters that may interfere with the use of Multicast. Although some routers come with the support to Multicast, their default settings avoid announcement of Multicast groups to devices outside the local network and the router will discard announcements arriving from outside. The devices can use the method internally but will be unable to communicate with devices at other institutions through Multicast.

Network administrators can enable the MSDP at these routers when the protocol is supported

by the network interface on the router. In this case, all devices in the route need to accept the same configuration. Routers or network interfaces may have or lack support for the necessary settings, and in the last case they need to be changed or the route need to follow another path to complete the tree of nodes.

At this point the users need to select and join a Multicast group before using it. This group is defined by an IP and port. The IP range for Multicast is defined on RFC1112 (Deering 1989) and goes from 224.0.0.0 to 239.255.255.255. The port number is expected to be an unassigned port in order to avoid problems with any other running service. The RFC6335 (Cotton *et al*, 2011) suggests the use of any dynamic port from 49152 to 65535 during tests or evaluations. The dynamic ports will never be assigned to any specific software by IANA[2] and the ports are expected to be free in most part of time.

Other network solutions for distributed communication are available for situations outside local or private networks. One of these technologies is the Cloud Computing and the Cloud Services, which are discussed in the next section.

3.5 Cloud Services

Once the devices are almost always connected to the Internet, solutions like Peer-to-peer, Ajax, sockets, and web services have been used to allow continuous data exchange. Although most of them present good performance, the advances on Cloud Computing provided support for many devices interconnection with on demand improvements regarding the capacity of services.

The implementation of services on the cloud is available by many companies like Amazon, Microsoft, Intel, IBM, and Google. On top of that we have companies that offer services already implemented on the cloud, make them ready to use, and take advantage of cloud features at the same time. The cloud services offered by these companies are commonly used today due to their facilities and performance. As the users have to use the services without access to the implementation, the evaluation of many cloud services may help to decide which one suits their needs.

Cloud services are services that are deployed on a cloud computing structure to take advantage of its computation and distribution qualities. Cloud Services differs in terms of message size, limit of connected devices, messages per second, and location of servers. Although these settings may be default in most cases, the companies have different plans that expand the possibilities and can also extend the capacities on demand depending on the situation. A common use is the data replication service at websites and mobile applications, so even if we have increasing access at some moment, the cloud service can instantiate another machine or machines to provide minimum latency and avoid processing overhead on the servers.

During this research some applications were developed using two specific cloud services: Pusher and PubNub. These cloud services were selected due to their popularity in the time of this research. Although they are quite similar, they implement specific rules to use their APIs that can interfere in some interaction approaches.

Pusher cloud service

Numerous mobile applications have a focus on fast message delivery to a high number of devices in many parts of the world. A good example of this service is an email application that sends us a notification whenever we receive a new message without requiring us to actively request new information. This approach is called push notifications, and one implementation of it is Pusher. This cloud service uses cloud computing solutions in order to offer a service that delivers messages through web sockets and HTTP streaming.

One of the advantageous features supported by Pusher API is its support of HTTP Keep-Alive feature. Once connected to Pusher cloud service, it is possible to send and receive messages to/from the cluster without starting a new connection, saving the overhead of restarting new TCP

[2]Internet Assigned Numbers Authority (IANA): http://www.iana.org (visited on May, 2017)

connections. The Pusher service offers the possibility of creating real-time applications considering all of its resources.

The service has restrictions for all plans, free and paid alike. Clients are allowed to send no more that 10 messages per second and will be disconnected from the service when they exceed this limit. This limitation is due to the overhead on distributing messages among a thousand users. Every message has a size limit of 10 kilobytes, but it is possible to request an upgrade in order to send larger messages.

The requirement to exchange messages between users is to create a channel and have users connected to the same channel. The API supports public, private, and presence channels. Public channels are used only to send messages from the server to the users connected, like a feed. Private channels need a prefix "private-", require server authentication, and offer the option to accept messages from its users. The presence channel is similar to the private channel, and includes the feature of requesting information about connected users. The channels may have different events that can be bound by clients, e.g. a client can bind the event "client-event" and receive notifications when a new event with the same name is sent to the channel. On private and presence channels, client events must have the "client-" prefix. An example code used to send and receive messages through the Pusher cloud service using a private channel is presented on Listing 3.1.

```
Pusher pusher;
PrivateChannel channel;

HttpAuthorizer authorizer =
  new HttpAuthorizer(AUTH_PAGE);
PusherOptions options = new PusherOptions();
options.setAuthorizer(authorizer);
pusher = new Pusher(PUSHER_API_KEY, options);
pusher.connect();

String channelName = "private-channel";
channel = pusher.subscribePrivate(channelName);
String eventName = "client-event";
channel.bind(eventName,
  new PrivateChannelEventListener() {...});
JSONObject jsonObject = new JSONObject();
String message = MESSAGE;
jsonObject.put("message", message);
channel.trigger(eventName, jsonObject.toString());
```

Listing 3.1: *Example of Java code from Pusher API*

In this piece of code, there are information that depends on the application settings. The first is the "AUTH_PAGE", that needs to be configured to give a unique authentication code to every socket connection. The "PUSHER_API_KEY" is the key received when creating an application on Pusher. After creating the channel, the programmer needs to bind an event with an event listener. The event listener requires the code that is expected to run every time the event occurs. The "MESSAGE" is a string with any values defined to be sent, e.g. numbers need to be converted directly to string or through Base64 binary-to-text encoding schema.

PubNub

PubNub cloud service provides an extensive API and SDKs, and the service can be easily configured with a few functions. Initially, the application needs to get an UUID and initialize the PubNub object including information regarding the account: the publish key and the subscribe key. After that, the device can publish messages to any channel as long as the channel name is known (e.g. "performer", "audience"). When subscribing to a channel, it is also necessary to define a callback function that will typically parse the received messages. The basic function to set up a publish-subscribe mechanism is presented on Listing 3.2.

```
// Request an UUID
var my_id = PUBNUB.uuid();
// Initialize with Publish & Subscribe Keys
var pubnub = PUBNUB.init({
 publish_key: publishKey,
 subscribe_key: subscribeKey,
 uuid: my_id,
});

// Subscribe to a channel
pubnub.subscribe({
 channel: my_id + ",audience",
 message: parseMessage, // callback for msg
 error: function (error) {
  // Handle error here
 },
 heartbeat: 15
});

// Parse received message
function parseMessage( message ) {
 if (typeof message.type !== 'undefined'){
  if ( message.type == "create-response"){
   // Do something
  }
 } else if ...
}

// Publish a message to a channel
pubnub.publish({
 channel: "performer",
 message: {"type":"create",
  "my_id":my_id,
  "nickname": strScreenName},
 error : function(m) {
  // Handle error here
 }
});
```

Listing 3.2: *Example of JavaScript code from PubNub API presented at audience page*

Pusher and PubNub can send any type of data through the cloud respecting plan limits. For instance, one might share codes from computer music languages like CSound[3], ChucK[4], and SuperCollider[5]. Cloud services can also share symbolic data or control signals between any mobile applications, allowing one to make use of any computer music synthesis engines available in tandem with Cloud services.

[3]CSound: http://www.csounds.com/ (visited on May, 2017)
[4]ChucK: http://chuck.cs.princeton.edu/ (visited on May, 2017)
[5]SuperCollider: http://supercollider.github.io/ (visited on May, 2017)

Chapter 4

Research Methodology

> Work it harder, make it better,
> Do it faster, makes us stronger,
> More than ever, hour after hour,
> Work is never over.
>
> Daft Punk

This research was conducted following quantitative and qualitative approaches. The quantitative approach evaluated the technologies selected for the experiments, while the qualitative approach gathered data from applications' users. Data was analyzed and used as basis for improvements of applications, based on users expectations.

The primary goal of this research, as stated in the first chapter, is to evaluate the current technologies and trends for mobile interaction, whereas a secondary goal is to promote the use of mobile technologies by musicians and artists, even without technical knowledge. These goals seek to bridge the gaps identified in mobile music practices, targeting those who intend to use network technologies and mobile devices for collaborative and cooperative interaction during music performances.

To accomplish these goals, applications were developed to provide working solutions for specific scenarios in mobile music, and experiments were conducted to map possibilities and difficulties found in such scenarios. The applications developed during this research have their description at Appendix E, following a chronological order that reflects the research process. Chapter 5 presents the network technologies evaluation along with discussions of the use of these technologies and identified constraints. Technology evaluation follows two patterns. Some evaluation target the automatic exploration of different parameters used in mobile music interaction, different services and routing schemes, and data gathered over long periods of time. Papers, talks, demonstrations, installations, and performances are the means to accomplish the secondary goal. Papers, published and presented at several computer music conferences in Brazil and abroad, offered opportunities to reach researchers, musicians, and lay audiences interested in mobile music. These papers were included in this thesis as Appendix F

During the development of this project, other related projects were developed by members of the computer music research group at IME/USP, whose methodologies also were considered in this study. André Bianchi's study of mobile platforms for realtime audio processing produced the *DSPBenchmarking* (Bianchi and Queiroz, 2012), an open source application for Android DSP evaluation. A partnership with André Bianchi allowed evaluations of DSP techniques that outperform traditional approaches, as presented in detail in Appendix E.2. Flávio Schiavoni developed *Medusa* (Schiavoni *et al*, 2011), an open source tool for network distributed performances with audio and MIDI. In Flávio's research, network tests using the loopback approach were conducted in order to calculate RTT between computers using different network transport protocols; this methodology was adapted to be used here with mobile devices and different services and routing schemes.

This thesis is also the result of an organic research methodology focusing on alternating periods

of studies, experimentation, and innovation. Periods of studies through courses and books were carried out before the first experiments with each technology addressed, leading to pilot experiments viewed as learning tools. Issues and alternatives that appeared during these experiments turned out to be valuable resources for periods of brainstorms leading to innovations in the strategies implemented. To propose innovative solutions, and to try them out by developing software, were stepstones of the research methodology that helped identifying gaps and shortcomings, which demanded new studies, leading to new research cycles.

The explorative investigation of technologies for mobile music may be categorized in two main branches or research plans. The first research plan focused on the evaluation of currently explored options from the mobile music state-of-the-art, as found in existing literature. Appendix E offers a comprehensive description of this first research plan containing the steps dealt with in detail. A second research plan is based on exploring options for mobile music from trending mobile technologies that are unpopular or little widespread in mobile music scenarios. These include newer technologies such as IPv6 and gigabit connections, that will be broadly available in the future but aren't yet at the time and place where this research was conducted, and also technologies that were available but didn't yet make their way into mobile music projects, such as Multicast routing and Cloud Services.

4.1 Methodology for exploring mobile devices in mobile music

The focus of this research are mobile devices, especially smartphones. Exploring mobile devices in real concert situations was the aim of many installations and performances conceived and carried out during this research process. Bearing this in mind, and also the availabity of specific devices within the collaborating research groups, most devices used during application development were Android-based, whereas iOS devices where used in specific partnerships. Other devices were also used in this research by audiences during web-based collaborative and distributed performances which would only require Internet access.

The Android operating system was selected as the main focus for application development during this research for some reasons, one of them being the ease of adopting FLOSS ideals in development. Applications were developed under free licenses with open sources and became available for use through Github repositories, F-Droid, and compiled applications. On one hand, source-based distribution is useful for collaborative development, and on the other hand distribution of compiled versions (apk) is more friendly to interested users and beta testers. Android also offers possibility of distributing applications through Bluetooth, WiFi, or web pages, without the burden of having to publish them or sign provisioning profiles with a device ID as in the case of iOS applications.

Android application development was based in two main IDEs. The Eclipse IDE was used during the development of the first applications, mainly due to its popularity among Android developers. This IDE requires the Android Development Tools (ADT) plugin for compiling applications and communicating with Android devices and emulators. By the end of the research period, the Eclipse ADT became deprecated and the application development moved on to Android Studio. Android Studio is the official IDE for the development of Android applications and replaced the Eclipse ADT by the end of 2015[1]. This IDE presents a complete environment offering many tools for debugging and monitoring running applications.

Emulators allowed running applications together with the IDEs inside the same machine. This option facilitates interface design and local network simulation. The emulators used during this research were based on the native images that come with the Android SDK, and also the Genymotion Android Emulator[2]. Although Android emulators can simulate virtual sensors at their current versions, some applications required the evaluation of sensors and other features unavailable at emulators during the development period and obliged the use of real devices as well.

[1] An update on Eclipse Android Developer Tools - Android Developers Blog: https://android-developers.googleblog.com/2015/06/an-update-on-eclipse-android-developer.html (visited on May, 2017)

[2] Genymotion Android Emulator website: https://www.genymotion.com/ (visited on May, 2017)

The mobile devices used during evaluations were LG D686 G Pro Dual Lite, Samsung GT-I9300 Galaxy SIII, and Sony Xperia D8533/D8503 Z3 Compact, abbreviated as D686, S3, and Z3, respectively. Their main technical specifications are presented on the Table 4.1, and the complete settings of the devices are described in Appendix C. Additionally, many uncatalogued personal devices were used during the application development process (by beta testers) and also during the installations (by the audiences).

Table 4.1: *Main technical specifications of the mobile devices used during evaluations: D686, S3, and Z3. Source: Adapted from GSMArena (2017)*

		LG D686 G Pro Lite Dual	Samsung GT-I9300 Galaxy SIII	Sony D8533 (and D8503) Xperia Z3 Compact
LAUNCH	Announced	2013, October	2012, May	2014, September
	Status	Available. Released 2013, November	Available. Released 2012, May	Available. Released 2014, September
PLATFORM	OS	Android OS, v4.1.2 (Jelly Bean)	Android OS, v4.0.4 (Ice Cream Sandwich), 4.3 (Jelly Bean)	Android OS, v4.4.4 (KitKat), upgradable to v6.0 (Marshmallow)
	Chipset	Mediatek MT6577	Exynos 4412 Quad	Qualcomm MSM8974AC Snapdragon 801
	CPU	Dual-core 1.0 GHz Cortex-A9	Quad-core 1.4 GHz Cortex-A9	Quad-core 2.5 GHz Krait 400
	GPU	PowerVR SGX531	Mali-400MP4	Adreno 330
MEMORY	Card slot	microSD, up to 32 GB (dedicated slot)	microSD, up to 64 GB (dedicated slot)	microSD, up to 256 GB (dedicated slot)
	Internal	8 GB, 1 GB RAM	16/32/64 GB, 1 GB RAM	16 GB, 2 GB RAM
COMMS	WLAN	Wi-Fi 802.11 b/g/n, Wi-Fi Direct, DLNA, hotspot	Wi-Fi 802.11 a/b/g/n, dual-band, Wi-Fi Direct, DLNA, hotspot	Wi-Fi 802.11 a/b/g/n/ac, dual-band, Wi-Fi Direct, DLNA, hotspot
	Bluetooth	v3.0, A2DP	v4.0, A2DP, EDR, aptX	v4.0, A2DP, LE, aptX
FEATURES	Sensors	Accelerometer, proximity, compass	Accelerometer, gyro, proximity, compass, barometer	Accelerometer, gyro, proximity, compass, barometer

Some applications for iOS devices using the *urMus* platform (Kim and Essl, 2011) were developed during the period as an exchange student at the University of Michigan under the supervision of Professor Georg Essl. Within Professor Essl's group many iOS devices, such as iPods, iPhones, and iPads, were available for research purposes. All of these devices had the *urMus* platform installed and this allowed the development of mobile music applications without the need of subscribing to the Apple Developer Program. A web interface for the *urMus* platform is available, which allows developers to deploy new codes and interfaces for mobile applications. The developed applications were tested on real devices and some of them were used during mobile performances by students. The main evaluation of these applications was made through qualitative reports during demonstrations and rehearsals, and also from feedback by Professor Essl before performances.

Some technologies such as PhoneGap[3] allow the development of hybrid solution using web technologies, where the same application runs on both Android and iOs devices without many modifications. During this research process, the main web technology used for hybrid application development was Audio Synthesis with Web Audio and network communication through Cloud Services. The compatibility of the Web Audio API with browsers available for several mobile platforms allows the participation of many users in a mobile music performance without the burden of having to install OS dependent applications, since all interaction and control is done within a web site.

4.2 Network settings

Intranet and Internet were used in order to evaluate network alternatives for mobile music interaction and collaboration. Some network communication options were available only in specific scenarios during the research and the applications evaluated through these solutions aimed to explore most of the available settings. The comparison of alternative mobile technologies was conducted under similar conditions whenever possible (i.e. when technologies could be employed in similar scenarios).

Intranet setting evaluations were related to protocol options and other technical aspects. Most of the evaluations using local networks were made using UDP for packet exchanging between Android devices and through Bonjour[4] for iOS. An additional evaluation was made using the network infrastructure available for academic institutions, which might be considered an Intranet due to its private access policy and backbone topology.

Academic institutions are usually connected through national networks which are in turn connected to other backbone networks in different countries. Brazilian National Research and Educational Network (RNP) manages the academic network available for Brazilian universities, while the Internet2 is responsible for interconnecting US universities, and RedCLARA is a network that interconnects Latin American academic networks. These interconnected backbone networks were used in this research in order to evaluate long distance communication between Brazil and US. This academic network allows the use of their services and also offers public IPs for devices. Due to their private access, the use of these networks was only made possible after registering the research project within the academic network managerial infrastructure.

Use of the Internet was an expected approach since this technology is widely accessible. Although many applications were designed for direct communication, some of them were designed to use Internet technologies such as web services and Cloud Services. These are interesting alternatives accessible to lay audiences and non-academic researchers as well.

Web service

A web service was considered as the first option for evaluating the use of solutions for mobile music interaction over the Internet, since using a web server for web service deployment is a common approach for web developers that want to distribute data between many devices and platforms. A web server allows the creation of an online application on a shared computer that can be reached by anyone connected to the Internet.

The traditional HTTP requests, GET and POST, are common options available in APIs developed for web services. During an interactive performance, the applications are expected to GET the most updated data and POST updated data in order to share with other devices, so the idea of focusing only on these requests was a natural approach for real-time interaction. Other requests such as UPDATE or DELETE are less useful in a real time context where data is immediately used after it is shared, unless the user need to manage previously exchanged data afterwards.

Web service development with an API can be hard for users without technical knowledge. Bearing this in mind, a Rapid Application Development (RAD) approach was an interesting alternative

[3] Adobe PhoneGap website: https://phonegap.com/ (visited on December, 2018)
[4] Bonjour website: https://support.apple.com/bonjour (visited on December 2018)

for exploration. The Ruby on Rails (RoR) framework[5] was selected to create the web application that would run on the web server and offer the required web services for exchanging data between mobile devices. This framework offers the possibility of creating applications with just a few lines of code, describing the data that will be exchanged and the fields and types of database elements. This framework is also an alternative for developers who intend to create complete multi-featured web applications without having to create their code from scratch based on a full Model View Controller (MVC) design pattern.

An application created with RoR offers ready-to-use requests for getting, posting, updating, and deleting data, including support for widely-used data formats such as JSON. This format was used in many applications developed during this research process due to its improved readability and cross-compatibility in comparison to alternatives such as XML.

The evaluation of the web service was done in local servers during development, but eventually the application was deployed on a shared web server. The Dreamhost[6] web-hosting plan was used in this case with a shared machine including two 4-cores AMD Opteron(tm) Processors (model 4122) with 32Gb of RAM. The Phusion Passenger[7] was the web server and application server installed in this machine in order to offer support for the RoR application. The setup of these technologies was based on user panels that required little effort in order to get at a complete ready-to-use solution.

Cloud Services

Cloud Services were also explored as an alternative to web servers. The former offers comparatively less control than the latter, but provides faster deployment and also scalability. This alternative is conceptually similar to a web server solution deployed in a distributed manner, since the Cloud is basically made of many interconnected servers. However, this interconnection structure is set up automatically, which offers some advantages for musicians and other professionals.

Cloud servers offer the possibility of automatic service replication through web interfaces, allowing an application to keep running even if many users decide to access it at the same time. Although most mobile application experiences in the literature expected the participation of only a few dozens of participants, the audience size can be extended with a scalable technology such as that of a cloud service. The evaluation of Cloud Services conducted in this research explored setup times, APIs and also the scalability levels of cloud solutions offered by some companies on free and paid plans.

Pusher and PubNub were selected for cloud evaluation due to the experiences during the development of the applications presented at the Appendix E. Both companies offer the possibility of using the cloud infrastructure without any burdens or setup of virtual machines. The main settings are available through the website of each company, whereas other settings are made available through their APIs. Although the companies offer demo keys for users aiming to evaluate the services without a commitment, the plans available provide private keys for secure data exchanging after the subscription process.

The cloud service APIs used here were designed for Android and Javascript. The methods were selected to simulate the GET and POST approach defined for web services, in order to facilitate comparison of both approaches. In this case, the Cloud Services present *publish* methods for posting data and *subscribe* methods for getting data. An important difference between the subscribe approach and the GET request is that, after subscription, all messages are delivered to all users subscribed to the same channel without any request from subscribers, making this approach lighter than GET. Personal keys allowed the use of private channels for communication during the evaluation.

The Cloud Service plans selected were the most affordable paid plan and the standard free plan available in both Pusher and PubNub. These plans had communication limits based on the number

[5] Ruby on Rails framework website: https://rubyonrails.org/ (visited on December, 2018)
[6] Dreamhost website: https://www.dreamhost.com/ (visited on May, 2017)
[7] Phusion Passenger website: https://www.phusionpassenger.com/ (visited on May, 2017)

Table 4.2: *Pricing and other details of selected plans from Pusher and PubNub Cloud Services as of 2017 (PubNub, 2017; Pusher, 2017).*

Service	Plan	Price in US$	Devices/Day	Messages		
				Quantity	Per Second	Size
Pusher	Sandbox	Free	100	200k/Day	10	32kb
	Startup	$49	500	1M/Day	10	32kb
PubNub	Free	Free	100	1M/Month	Unlimited	32kb
	Growth Tiers	$49	500	2M/Month	Unlimited	32kb

of connected devices and the number of exchanged messages per day. A description of these plans is presented in Table 4.2.

Although the limits of the paid plan and the free plan are similar, these limits are high enough to accommodate traditional mobile music performances as found in the literature. The idea of comparing both plans was to figure out whether the paid plan had technical advantages regarding the Cloud technologies. From the developer point-of-view, the only coding difference between the free or paid plans was the key used to connect to the Cloud Service platform (i.e. there is no significant difference). The paid plan was used during actual performances to avoid problems with limits imposed by the free plan.

Applications created using Cloud Services focused on performance and network evaluation. PubNub was used in the pubslides application described in Appendix E.6, and in "Crowd in C[loud]" application/performance described in Appendix E.8. Pusher was used in SuperCopair application/performance described in Appendix E.7. The *PushLoop* application described in Section 4.3 was developed to evaluate these Cloud Services free and paid plans, compare their performance with other network alternatives simulating real time interaction during performances.

Protocols and Routing

Cloud Services were compared to network alternatives, considering several protocols and routing methods. The protocols used during this research process were UDP, IPv4, and IPv6, while the routing methods were Unicast and Multicast. These options were selected as a baseline reference since they are supposed to be ready-to-use by devices connected to an academic network.

These alternatives offer limits regarding bandwidth throughput and number of connected devices depending on the specific network. At the University of São Paulo the network managers defined a IPv4 subnet with a CIDR (Classless Inter-Domain Routing) */29*, a range of 8 IPs, and a IPv6 subnet with a CIDR */64*, a range of 18,446,744,073,709,551,616 IPs (a quintillion of IPs). The IPv4 setting was more restrict than IPv6 and allowed only 5 devices to be connected at the same time, while the other 3 IPs being reserved for gateway, router and broadcast. At the University of Michigan the devices were registered on the network using their MAC addresses with IPs specified by the network managers. In both universities devices were connected to the network through specific VLANs that offered public IPs with Intranet and Internet access.

Network constraints evaluation

During the network evaluation, performance constraint simulations had predefined parameters. The number of packets per second, or packet rate, varied by introducing a delay between packets of up to 1000 ms, and packet sizes varied from 1 to 250 32-bit floating-point values. These parameters were taken to represent typical rates related to mobile sensors and other inputs such as touch events, either in single events or grouped events. These values were used for the comparison of the network alternatives described above, as detailed in Section 4.5.

4.3 Network evaluation on Android: PushLoop

During this research process, many network technologies were evaluated through a mobile app. The primary purpose of this application is to provide an easy way to exchange messages between mobile devices around the world, and evaluate the communication options. The application created for this evaluation was named PushLoop, and included settings for Unicast IPv4, Unicast IPv6, Multicast, and Cloud Services.

PushLoop Android application is based on the true echo concept, where one device sends messages (the "*PUSH*"), while the other device (the "*LOOP*") answers the message as soon as possible, which is a single reflection of a source. The first device registers all messages sent and received on a report file with millisecond time-stamp precision. The method used to register events on the report was *SystemClock.elapsedRealtime()*. We also run AsyncTask invoking *executeOnExecutor()* with *THREAD_POOL_EXECUTOR* in order to have parallel execution on Android. The messages have five components: a unique device id; message number with cycle number on the range of thousands; the total number of messages; a random integer as a key for message identification; and a block of floats. An example of a message used during the tests is presented below:

23.0.1.A87422113 1001 1000 76 [0.28506452]

An activity diagram is presented in Figure 4.1, representing the point of view of a sender activity on the application. The class diagram is presented in Figure 4.2. This diagram includes all class from the current version of the application after many changes during the development process due to network experimentation. Screenshots of the PushLoop application are presented in Figure 4.3.

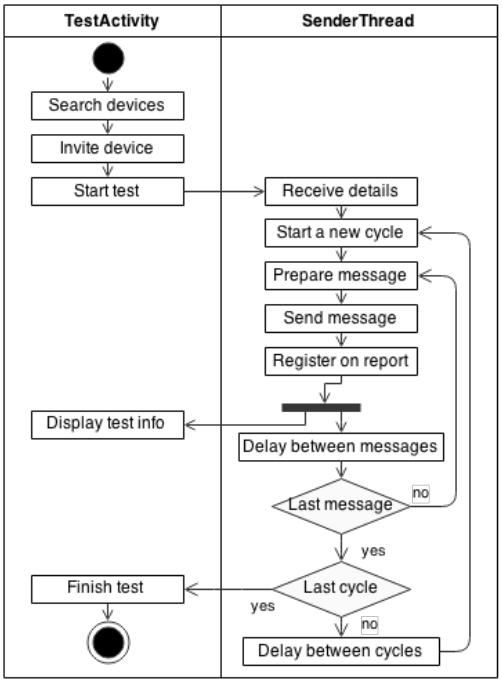


Figure 4.1: *PushLoop activity diagram of the test from the sender point of view.*

The code of the application is available at https://github.com/deusanyjunior/PushLoop (visited on May, 2017). The first evaluation using this application was published as a paper at the International Computer Music Conference, at Denton, Texas, USA (de Carvalho Junior *et al*, 2015b). This paper is presented in Appendix F.7. Chapter 5 is based on the evaluations conducted with this application.

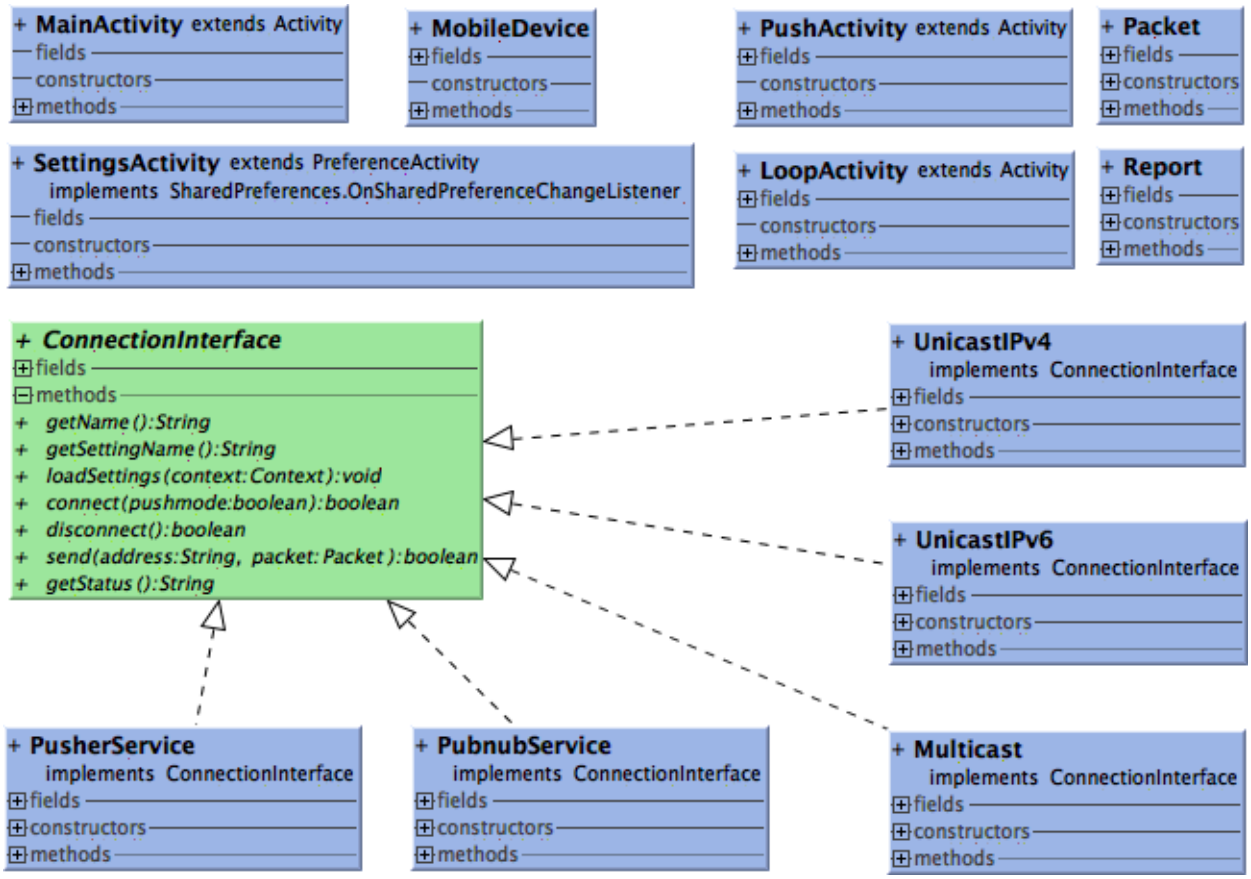
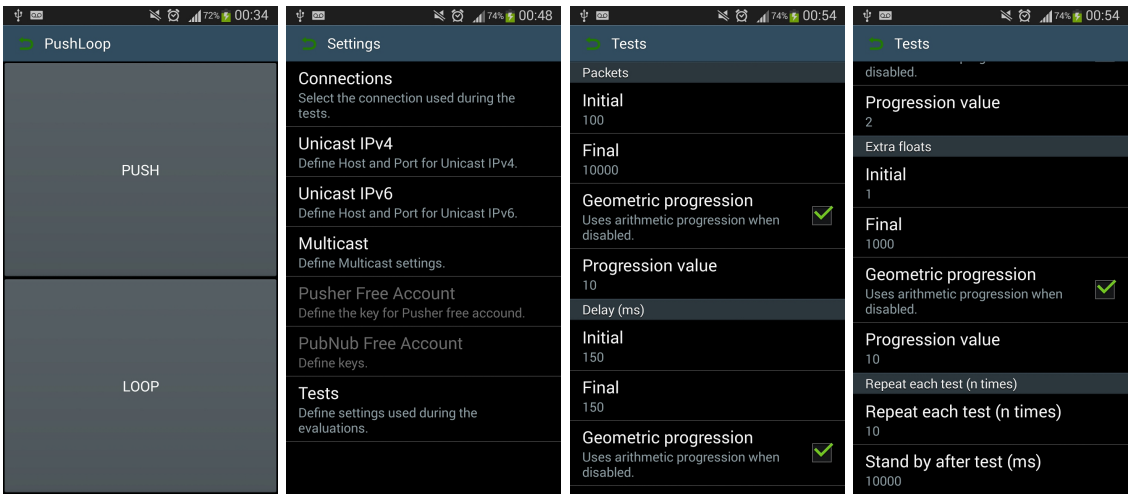


Figure 4.2: *PushLoop class diagram.*



(a) *Main screen* **(b)** *Setting screen* **(c)** *Test screen (1)* **(d)** *Test screen (2)*

Figure 4.3: *PushLoop application screenshots.*

4.4 Music Processing

The mobile music scenarios addressed in this research always considered that audio synthesis and processing took place within mobile devices, and that network communication was mainly used for symbolic and control data. The reasons for this assumption is twofold: on the one hand, previous work at the Computer Music research group at IME/USP already indicated that most Android devices back in 2012 were able to process audio in real-time (Bianchi, 2014); on the other hand, latency and jitter in network audio transmission were important factors for mobile devices (mostly

connected wirelessly), together with transmission costs due to mobile data plans that also rendered audio streaming less appealing in scenarios with large audiences. Some programming languages specifically designed for music and audio processing running inside mobile devices were at the core of many applications proposed here.

Pure Data is the main computer music language used in this research, and is used in the applications described in Appendices E.1, E.4, and E.5. The CSound language was used for the development of the application *Touches On The Line*, described in Appendix E.3. SuperCollider was used in the development of the collaborative and cooperative live coding application named *SuperCopair*, which is presented in Appendix E.7.

The applications developed for mobile music interaction through a web browser were created using the Web Audio API for audio synthesis. Appendices E.8 and E.10 describe the applications *Crowd in C[loud]* and *Open Band*, respectively, that use Web Audio for music processing.

During the development of applications for iOS devices, the *urMus* platform was used, where audio synthesis is done using the STK. Music synthesis can be programmed using flowboxes in the *urMus* interface or with Lua language flowboxes abstractions.

4.5 Scope of the Evaluation Process

Technologies were improving during this research, and while some technologies were getting increasingly popular, such as the 802.11ac wireless standard and Cloud Services, others were hardly used since its conception, such as Multicast data transmission over academic networks and IPv6 sockets. Some selected technologies were a bet due to the uncertainty of the market, with an expectation that these technologies would become popular in a near future.

Sensors and packet rate

Android Developer Guide for Sensors[8] describes four different sensor rates named as Normal (with a delay of 200 ms between each sample), UI (with 60 ms), Game (with 20 ms), and Fastest (with 0 μ s, that means the device will get sensor data as fast as possible). From the experiences with the development of some applications such as the Thereminal (Appendix E.1) and Sensors2PD (Appendix E.4), the users noticed that most sensor rates work as expected, except the Fastest rate, which displayed an unstable rate. An evaluation of Android sensors determined the minimum delay between sensor events to be around 5ms for accelerometer at fastest sampling rate (Ma *et al*, 2013).

These values express that the maximum frequency is 200 Hz for three float numbers representing the accelerometer coordinates. For instance, the touch responsiveness of Android devices depends on the screen scan frequency but some circuits can scan at 120 Hz and find up to 10 fingers position, what implies 10 pair of position values on every 8 ms in the best situation (Padrę, 2017). Bearing in mind these values, this research process considered the sampling rate varying from 4 Hz to 500 Hz, in order to verify the performance at normal situations and also at fastest ones.

Packet size

According to the RFC1191 (Mogul and Deering 1990), the Maximum Transmission Unit (MTU) is 1500 bytes in most Ethernet Networks. It means that a packet will be fragmented in case its size is higher than this unit, and the device will need to wait more than one packet to get the full data and interpret the information. In that sense, the packets sent during the evaluations were defined to carry less than 1500 bytes of data in order to avoid packet fragmentation.

[8] Android Developer Guide for Sensors: https://developer.android.com/guide/topics/sensors/ (visited on December, 2018)

WiFi communication

The WiFi connection was set-up at AC1750 to work only with 802.11a and 802.11ac standards, reaching up to 1Gbps. Although house clients in Brazil have Internet connection available in Mbps, companies and Brazilian universities can take advantage of Gbps connections (up to 50Gbps). In the USA, Gbps connections are already offered as home Internet services, and Universities have 100Gbps available. Considering that the Internet options are getting better with time, the use of gigabit connections in this research is a bet for future MM proposals.

Unicast and Multicast

Multicast and Unicast were selected to interconnect devices within academic networks in order to take advantage of the predefined routes. In order to use Unicast, the user needs to know the public IPs of all devices that will exchange messages. Although public IPv4 is a scarce network resource today, IPv6 is becoming available for more devices.

Cloud Services

Cloud Services interconnect any device compatible with their APIs. Pusher and PubNub were the Cloud Services selected for this evaluation, and Pusher presented some restrictions such as 10kB per message and a maximum rate of 10 messages per second per device, while PubNub restricted message exchanging to 32kB per message (but devices can send as many messages as they want). The size of messages was above the MTU size, but the limit of messages per second of Pusher implied a minimum delay of 100ms between messages during the tests including Pusher.

Although we had few steps to set up the cloud services, the limit of messages to be sent each day imposed limits to the whole evaluation process. As long as we planned many evaluations and we would need to evaluate all services with the same settings in a sequence in order to share similar network conditions, it was necessary to divide the number of evaluations per day depending on the number of messages sent by the cloud service free plan.

The free account at Pusher had an unchangeable configuration that would stop exchanging messages after reaching the daily limit. It is important to remember that PubNub service has limits on the free plan, but the system avoids blocking the service and sends emails instead (these e-mails included information about the number of messages exchanged and invited the user to change the plan).

Routes

The routes between universities and the possible location of the cluster used by the Cloud services are presented in Figure 4.4. It is important to notice that every message needs to be sent to the cluster before being pushed to its final destination. This means that every message exchanged between São Paulo and João Pessoa passes by the cluster in USA before reaching its destination. As one of the main route to USA from João Pessoa through the RNP network passes by São Paulo, it is also possible that every packet in a route JPA-SAO passes twice through São Paulo backbone before reaching its destination inside USP.

The routes are undefined from the point of view of the application, but the packet follow the rules available on the routers and backbones. In case of using Multicast, it is necessary to inform every network manager to set up the rule for this kind of communication. As this research process interconnected universities from Brazil and USA, network managers from RNP (Brazil), RedCLARA (Latin America), and Internet2 (USA) were contacted lots of time in order to fix the configuration at their side before the experiments.

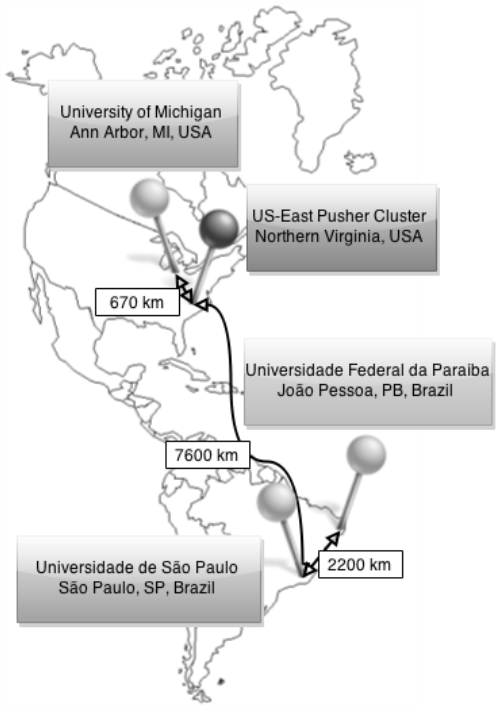


Figure 4.4: *Routes between the universities with linear distance in kilometers*

Evaluation summary

As this research aimed at many different types of evaluations around the same theme, Mobile Music Technologies, the results for each evaluation and their discussion are presented separately. The main division of the evaluations is related to the technical or musical aspects. Although some technical aspects were also discussed in the published papers, the network communication aiming music interaction is described in depth in Chapter 5 following the scope presented in this Section. Applications, performances, and music synthesis technologies are briefly discussed in Appendix E, and an extensive discussion about them is presented in Appendix F, which contains the papers published in conferences during the research process.

In a nutshell, the evaluation discussed in the next Chapter is based in the following specifications presented below:

- The research involved: the University of São Paulo (USP) at São Paulo, SP, Brazil (SAO), the Federal University of Paraíba (UFPB) at João Pessoa, PB, Brazil (JPA), and the University of Michigan (UMich) at Ann Arbor, Michigan, USA (ARB);
- The academic gigabit research network infrastructure was used in this evaluation both in Brazil and USA, being the Rede Nacional de Ensino e Pesquisa (RNP) and Internet2 (I2) responsible for the interconnection within both countries, respectively, while the RedCLARA is responsible for intermediate connection paths between RNP and I2.
- LG D686 G Pro Dual Lite (D686), Samsung GT-I9300 Galaxy SIII (S3), and Sony D8533 (and D8503) Xperia Z3 Compact (Z3) were the mobile devices used during the evaluations;
- The wireless router TP-Link AC1750 Archer C7 Wireless Dual Band Gigabit Router (AC1750) used at USP and UMich was set up exclusively for this research, while the AirPort Time Capsule used at UFPB was the available router for users at the Digital Video Applications Lab (LAViD). Both routers have gigabit connection and are compatible with the 802.11ac wireless standard;
- The interconnection between devices was managed through Multicast, Unicast, and Cloud Services;

The technologies used during the evaluations related to packet networking are presented in Table 4.3. The evaluations are separated in three experiments with different specifications. Their specifications are presented in Tables 4.4, 4.5, and 4.6. These evaluations were managed by the PushLoop application and executed on Android devices. The results from the First experiment is also discussed in the paper presented in Appendix F.7.

Network	
Communication	**Protocols**
Pusher	UDP
PubNub	IPv4
Unicast	IPv6

Table 4.3: *Summary of Network technologies evaluated in Chapter 5.*

Routes	**Communication**	**Delay (ms) Between Messages**	**Messages Size (number of floats)**
SAO-JPA	Pusher	150	1
ARB-JPA			50
			100
			150
			200
			250

Table 4.4: *Summary of technical specifications evaluated during the First experiment presented in Section 5.1.*

Routes	**Communication**	**Delay (ms) Between Messages**	**Messages Size (number of floats)**
SAO-JPA	Pusher	150	1
ARB-JPA	Pusher Paid		50
ARB-SAO	PubNub		100
	PubNub Paid		150
	Unicast		200
			250

Table 4.5: *Summary of technical specifications evaluated during the Second experiment presented in Section 5.2.*

The papers in Appendix F discuss the evaluation of the technologies presented in Table 4.7. Other technologies were evaluated during the development of the applications and they are presented in Table 4.8 and discussed in Appendix E.

Routes	Communication	Delay (ms) Between Messages	Messages Size (number of floats)
SAO-ARB	Unicast IPv4	2	2
	Unicast IPv6	4	4
		8	8
		16	16
		32	32
		64	64
		128	128

Table 4.6: *Summary of technical specifications evaluated during the Third experiment presented in Section 5.3.*

Network		Audio Synthesis
Communication	**Protocols**	
Web Servers	UDP	Pure Data
Cloud Services	IPv4	Csound
Unicast		Super Collider
		Web Audio

Table 4.7: *Summary of technologies evaluated in the papers published during the research and presented in Appendix F.*

Network		Audio Synthesis
Communication	**Protocols**	
Web Servers	UDP	Pure Data
Cloud Services	mDNS	Csound
Unicast		Super Collider
Multicast		Web Audio
Broadcast		urMus
Bonjour/Zeroconf		

Table 4.8: *Summary of technologies evaluated in the applications created during the research process and presented in Appendix E.*

Chapter 5

Evaluations, Results, and Discussion

> Buy it, use it, break it, fix it,
> Trash it, change it, mail - upgrade it,
> Charge it, point it, zoom it, press it,
> Snap it, work it, quick - erase it..
> ..
> Technologic. Technologic.
>
> Daft Punk

The context of the evaluations in this chapter follows the proposals and results from past research from Compmus group members. First, a collaboration with André Jucovsky Bianchi resulted in an evaluation of realtime DSP on Android devices exploring the advantages of using JNI instead of pure Java (de Carvalho Junior *et al*, 2013). In his thesis, André concluded that most Android devices are able to process audio in realtime even for large blocks of samples (Bianchi, 2014). The focus of the following experiments is the performance of mobile device technologies related to data transmission using different network alternatives for long distance interaction. Particularly, the evaluation focuses on the transmission of symbolic data (i.e. synthesis and effects parameters, control data, and sensors values).

5.1 First experiment: Size of packets

The first experiment evaluated the network communication through the Pusher cloud service using a mobile application. This evaluation was conducted between Ann Arbor, São Paulo, and João Pessoa in December, 2014, and January, 2015. Some pilot tests performed before the full evaluation had an intermittent loss of connection when exactly 10 messages were sent per second, which represent the nominal maximum allowed. In order to avoid this issue a 150 ms delay between messages was used, and we established 10 cycles of 100 messages per test with a delay of 500 ms between each cycle during the tests. Each test represented a performance of approximately 3 minutes with these definitions.

Six tests were conducted based on the number of floats sent as arguments inside the messages. We selected messages with 1, 50, 100, 150, 200, and 250 floats. In this case, the evaluation simulated messages in a range from 7 to 1750 floats per second.

The geographic configuration imposes physical restrictions on the lower bounds of network performance. We made use of the broadband Internet connection between the universities, whose actual interconnection medium is an optical fiber, with a light traveling speed of about 200,000 km/s. The cloud service main cluster is situated at Northern Virginia (NVG) in USA[1]. The Internet connection between João Pessoa (JPA) and abroad has three intermediary nodes in the path inside RNP: Fortaleza (FOR), São Paulo (SAO), and Porto Alegre (POA)[2]. The route passing through

[1] Information about Pusher cluster: https://pusher.com/docs/clusters (visited on November, 2018)
[2] RNP routes: https://www.rnp.br/servicos/conectividade/rede-ipe (visited on November, 2018)

SAO has the highest throughput, and was selected for the next measurements. Additionally, all messages should pass through Pusher's main cluster. Considering the distances and routes on the map in Figure 4.4, we have a theoretical (light speed) RTT of 174ms in the first evaluation, between São Paulo and João Pessoa (SAO-JPA)[3], and of 104ms in the second evaluation, between Ann Arbor and João Pessoa (ARB-JPA)[4].

Table 5.1 and Table 5.2 show a summary with the main results extracted from all measurements. The results summarized in Table 5.1 present a minimum RTT of 329ms with 150 floats and average RTT values between 486-578ms for all message sizes. On the other hand, we have a minimum RTT of 166ms with 1 float in Table 5.2, and average RTT values between 230-348ms. We sent 1000 messages on each test and lost no more than 4% of the messages in the worst case. Packet loss and RTT were higher in SAO-JPA link probably due to the distance, as the cluster is situated in USA. Moreover, ARB-JPA evaluation had runs with zero message loss.

The actual RTT for each message sent is presented in the charts in Figure 5.1. Some values overshoot the chart limit (chosen to maintain the scale of all charts), and they are considered to be lost during transmission (i.e. either a packet loss or a broken connection). The SAO-JPA evaluation presented some instability regarding the average RTT and there was no discernible pattern linking the number of floats to the average RTT. Although we have had some high RTT values during the tests, the concentration of high RTT values appear to be more frequent in clustered sequential messages than when messages are isolated (meaning musical algorithms based on sporadic communication would suffer less). Furthermore, it was relatively common for the service to lose at least one message after a high RTT value.

The RTT was somewhat stable during this experiment as visually presented in Figures 5.2 and 5.3. However, the tests between ARB and JPA seems to indicate a correlation between the large messages and high RTTs for this path. These results are also discussed in the paper presented in Appendix F.7.

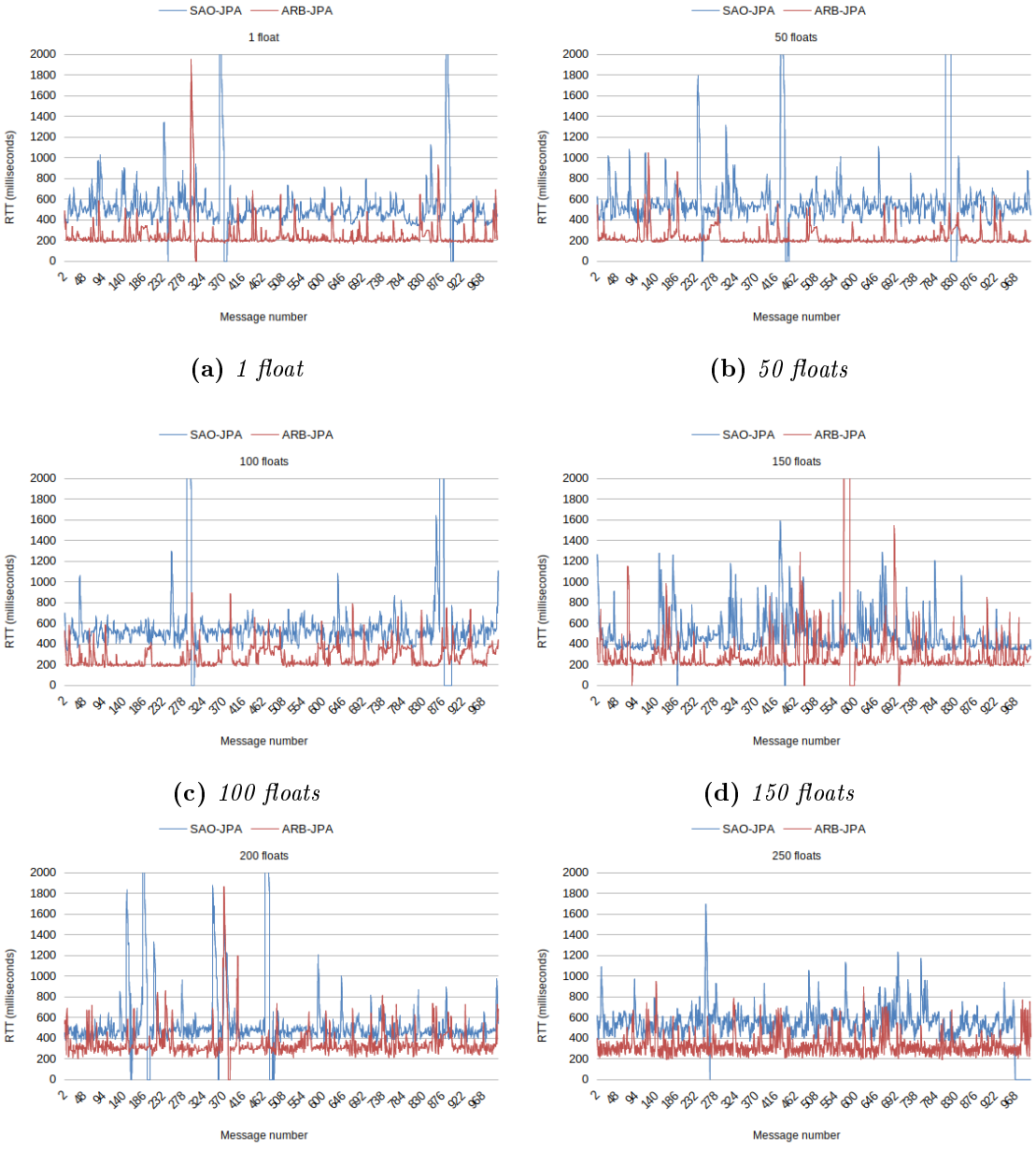
		Floats per message					
		1	50	100	150	200	250
Message	Loss (for 1000)	14	26	25	3	21	38
	Size (bytes)	41	614	1190	1782	2355	2950
RTT (milliseconds)	Minimum	342	332	332	329	332	352
	Maximum	2430	3916	4371	1595	3014	1700
	Average	515	578	563	486	536	543
	Standard deviation	224	366	394	181	305	168

Table 5.1: *Results from RTT evaluation using cloud services between SAO and JPA.*

		Floats per message					
		1	50	100	150	200	250
Message	Loss (for 1000)	3	0	0	17	5	0
	Size (bytes)	43	613	1189	1784	2378	2935
RTT (milliseconds)	Minimum	166	172	172	182	199	190
	Maximum	1953	1052	898	3100	1869	951
	Average	243	230	273	316	348	329
	Standard deviation	138	83	103	317	143	101

Table 5.2: *Results from RTT evaluation using cloud services between ARB and JPA*

[3]The distance between São Paulo and João Pessoa is 34,800km (route SAO-NVG-SAO-JPA-SAO-NVG-SAO).
[4]The distance between Ann Arbor and João Pessoa is 20,940km (route ARB-NVG-SAO-JPA-SAO-NVG-ARB).



(a) *1 float* **(b)** *50 floats*

(c) *100 floats* **(d)** *150 floats*

(e) *200 floats* **(f)** *250 floats*

Figure 5.1: *RTT comparison for different packet size sent from SAO to JPA and ARB to JPA. Lost messages are represented with 0ms RTT. The y-axis maximum is 2000ms for better visualization, but there are some RTTs reaching 4s.*

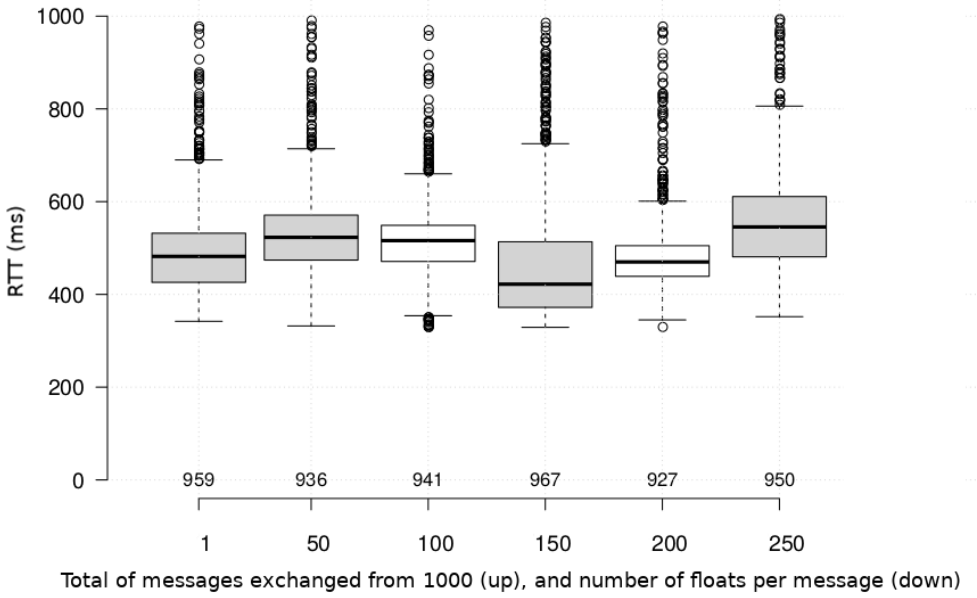


Figure 5.2: *Boxplot of RTT evaluation with Pusher cloud service and different message sizes between SAO and JPA. Delay between packages fixed in 150 ms*

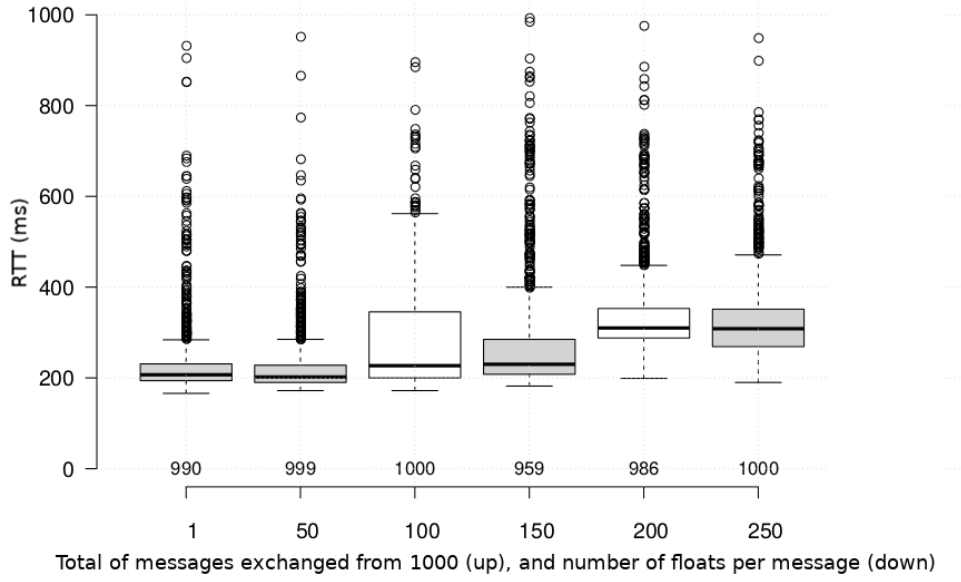


Figure 5.3: *Boxplot of RTT evaluation with Pusher cloud service and different message sizes between ARB and JPA. Delay between packages fixed in 150 ms*

5.2 Second experiment: Delay between packets

Based on the results of the first experiment we decided to evaluate more services changing the size of the packets. This evaluation was conducted between Ann Arbor, São Paulo, and João Pessoa in June and July, 2015. In this case, we evaluated the Pusher and PubNub cloud services regarding their free and paid plans, and we included the Unicast as a reference. The tables below present the results of the evaluations including these services and the statistical evaluation of the RTT results. The Pusher paid plan is named pusherP, while the PubNub paid plan is named pubnubP. In the tables' headers, Nobs is the number of observations, and when this number is under 1000 it means the we had packet loss during the evaluation. The size is the number of float numbers inside the packet sent.

Evaluation between São Paulo and João Pessoa

Table 5.3 presents the results for RTT evaluation with all the services between SAO and JPA. The results from PubNub were better than the results with Pusher in most cases. The possible reason is because PubNub has a cluster in Brazil. In this case, the packets from PubNub had a shortest path. The packets sent by Pusher had to travel from São Paulo to USA before coming to João Pessoa, and they also traveled to USA from João Pessoa before coming back to São Paulo.

Figure 5.4 presents the mean RTT between SAO and JPA considering only 150 ms delay between packets, in order to compare with the first experiment. The results were similar for Pusher, even for paid plan and in a different period of the year. As discussed before, PubNub had better results, but Unicast surpassed all other options as expected. The apparent parallelism between the results from PubNub and Unicast also implies some similar routes.

The experiments were done automatically by the PushLoop application and they were later analyzed. This approach had an impact on the final results as the unexpected results could be better studied during the time of its occurrence by means of application notifications. As for example, the packet loss during some tests could be reported to the user in the interval between tests. Also, the users could define a threshold for RTT based on terrestrial distances and set up a notification for RTTs exceeding this limit. In this evaluation, the results for PubNub with 1 float per message could be benefit from these notifications as the results are far from the others, although it is just an assumption and requires better analysis.

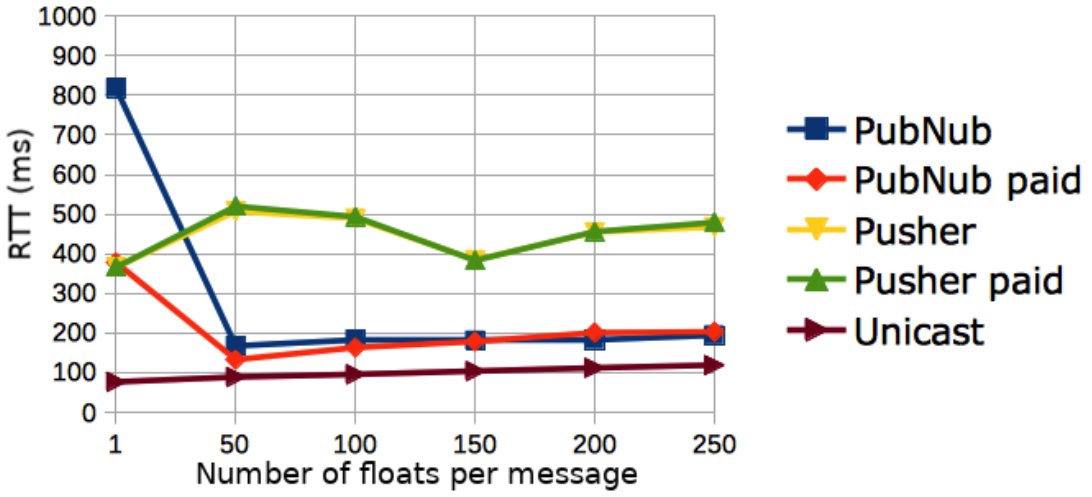


Figure 5.4: *Mean RTT between SAO and JPA for different services and message sizes. Delay between packages is 150 ms*

Table 5.3: *Second experiment results: RTT evaluation between SAO and JPA*

Service	Size	Delay	Nobs	μ	σ	σ^2	Skew	Kurt	Min	q1	q2	q3	Max
pubnub	1	150	**999**	**2641**	1436	2063440	1.64	8.04	**424**	1413	2471	3795	**13622**
pubnub	50	150	1000	201	251	63133	4.19	19.29	79	110	123	154	1983
pubnub	100	150	1000	187	139	19391	3.66	14.90	89	129	145	175	1155
pubnub	150	150	1000	230	359	129336	7.02	55.76	99	138	154	188	4060
pubnub	200	150	1000	188	120	14373	5.47	36.94	104	143	158	192	1405
pubnub	250	150	1000	196	120	14517	4.53	23.56	108	146	166	198	1228
pubnubP	1	150	**993**	400	194	37648	3.88	21.49	105	303	360	430	2122
pubnubP	50	150	1000	143	143	20451	8.11	76.99	81	107	115	130	1917
pubnubP	100	150	1000	189	227	51714	5.83	39.37	86	124	136	156	2374
pubnubP	150	150	1000	210	232	54110	4.95	27.65	94	137	148	178	2205
pubnubP	200	150	1000	238	264	69520	4.21	21.17	103	139	156	195	2368
pubnubP	250	150	1000	233	251	62988	5.16	31.53	108	148	164	200	2434
pusher	1	150	1000	367	56	3128	5.58	47.02	311	341	353	373	1049
pusher	50	150	1000	509	135	18189	0.86	1.01	321	395	490	603	1180
pusher	100	150	1000	490	87	7626	1.01	3.80	327	427	487	545	1047
pusher	150	150	1000	384	63	3948	4.71	30.13	322	355	369	392	939
pusher	200	150	1000	455	57	3282	3.36	28.72	343	426	447	477	1139
pusher	250	150	1000	471	82	6675	4.07	34.96	341	427	465	499	1410
pusherP	1	150	1000	366	37	1363	3.30	21.47	316	344	357	376	759
pusherP	50	150	1000	527	142	20087	1.29	3.89	325	416	513	610	1426
pusherP	100	150	1000	495	92	8430	1.30	6.36	332	430	493	549	1204
pusherP	150	150	1000	385	71	5097	5.71	45.45	323	353	369	392	1205
pusherP	200	150	1000	457	66	4312	4.03	34.89	352	423	445	477	1276
pusherP	250	150	1000	482	85	7293	3.25	23.24	346	434	471	518	1358
unicast	1	150	1000	**77**	47	2207	11.69	157.20	**59**	68	71	73	860
unicast	50	150	**997**	89	46	2160	9.51	106.26	63	79	83	86	722
unicast	100	150	**999**	96	34	1181	8.84	112.41	66	84	91	95	664
unicast	150	150	**997**	104	31	989	2.79	9.90	68	88	94	102	352
unicast	200	150	1000	112	51	2624	7.42	89.43	68	92	97	108	911
unicast	250	150	1000	119	58	3425	7.45	79.81	77	96	101	118	948

Evaluation between Ann Arbor and São Paulo

Table 5.4 presents the results of RTT evaluation with all the services between Ann Arbor and São Paulo. In this evaluation the PubNub had RTT higher than expected, but with better results on the paid plan. These results are possible due the Service Layer Agreement (SLA) of 99.999% for paid plan users [5]. The PubNub and Pusher had similar results between the free and paid plan. It is plausible that all traffic passed through the same clusters for both plans in this case.

Figure 5.5 shows the mean RTT during the evaluation between ARB and SAO with 150 ms delay between packets. The lines for Pusher and Unicast got similar results to previous tests, but PubNub results were somewhat unexpected. In this case, Unicast and Pusher were closer than before, and the results still seem to provide clues that the number of floats per message has a correlation with the RTT. PubNub were somewhat stable but higher than expected for this path.

[5] PubNub SLA support page: https://support.pubnub.com/support/solutions/articles/14000043627 (visited on December, 2018)

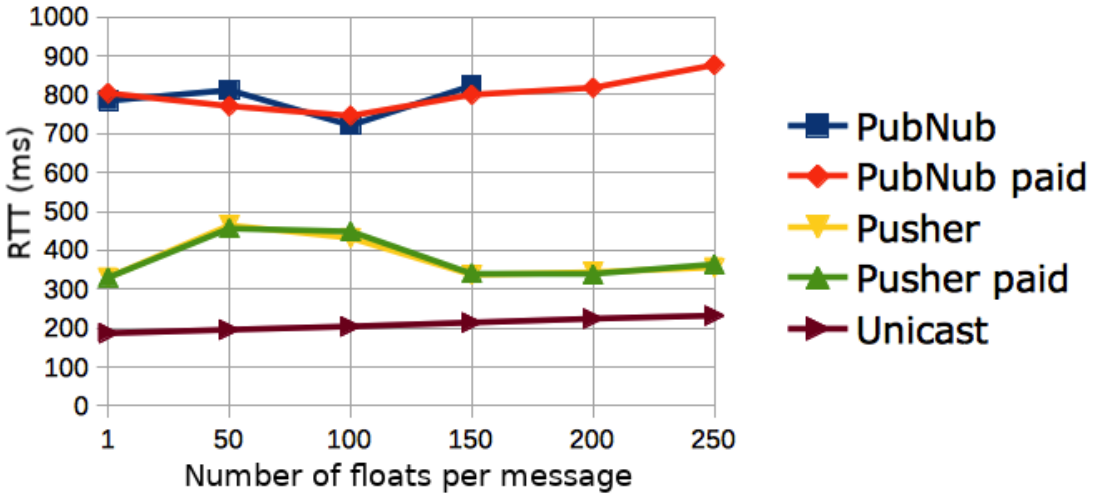


Figure 5.5: *Mean RTT between ARB and SAO for different services and message sizes. Delay between packages is 150 ms*

Table 5.4: *Second experiment results: RTT evaluation between ARB and SAO*

Service	Size	Delay	Nobs	μ	σ	σ^2	Skew	Kurt	Min	q1	q2	q3	Max
pubnub	1	150	1000	4486	2394	5,735,159.30	-0.32	-1.09	533	1785	4808	6865	11925
pubnub	50	150	1000	2802	2169	4,707,652.22	0.66	-1.40	411	1184	1593	5463	7452
pubnub	100	150	1000	8373	3949	15,609,162.36	-0.22	-1.18	574	5002	8961	12745	14132
pubnub	150	150	1000	12413	6835	46,760,628.81	0.06	-1.28	691	5632	12234	18208	24622
pubnub	200	150	1000	**15671**	7297	53,303,033.53	-0.09	-1.26	**2512**	9251	15600	22953	**27587**
pubnub	250	150	1000	12653	6277	39,441,152.32	0.00	-1.25	1710	6883	12078	18890	23102
pubnubP	1	150	1000	1001	337	114,005.71	0.79	-0.24	382	764	926	1105	1963
pubnubP	50	150	1000	2430	821	675,257.35	-0.71	-0.38	412	1982	2588	3052	3753
pubnubP	100	150	1000	9846	6167	38,072,365.22	0.09	-1.42	503	3318	10591	15964	19432
pubnubP	150	150	1000	9705	5531	30,620,190.96	0.39	-0.95	661	4639	8725	13502	19989
pubnubP	200	150	1000	12396	6090	37,121,911.71	-0.24	-1.29	711	7128	13680	17999	21050
pubnubP	250	150	**948**	12742	7234	52,385,134.53	-0.07	-1.11	697	5970	14169	18398	26510
pusher	1	150	1000	330	71	5,059.07	2.64	8.26	252	292	306	329	756
pusher	50	150	1000	465	119	14,274.06	0.86	0.67	268	383	446	532	938
pusher	100	150	1000	432	94	8,814.99	0.77	1.97	271	348	438	483	984
pusher	150	150	1000	337	69	4,780.45	3.11	13.82	261	297	318	346	922
pusher	200	150	1000	344	64	4,118.55	2.75	10.34	259	307	325	356	777
pusher	250	150	1000	356	74	5,546.34	2.72	9.92	258	313	334	366	873
pusherP	1	150	1000	330	69	4,833.24	2.43	7.60	251	290	305	342	754
pusherP	50	150	1000	456	113	12,726.50	0.74	0.33	255	366	446	514	867
pusherP	100	150	1000	448	86	7,480.51	0.47	1.13	264	401	447	498	868
pusherP	150	150	1000	347	107	11,495.46	5.28	35.46	261	301	319	348	1341
pusherP	200	150	1000	338	51	2,590.98	2.33	8.29	264	306	324	351	704
pusherP	250	150	1000	364	75	5,623.51	2.09	7.04	269	314	337	400	867
unicast	1	150	1000	**187**	38	1,425.68	4.08	19.14	**162**	173	175	179	457
unicast	50	150	**998**	196	34	1,162.23	4.23	20.61	167	183	188	191	458
unicast	100	150	**999**	204	33	1,101.55	4.62	26.92	171	191	197	202	503
unicast	150	150	**999**	215	46	2,079.68	7.26	99.10	174	196	204	210	1013
unicast	200	150	1000	224	42	1,732.19	3.07	12.20	175	203	211	221	511
unicast	250	150	**999**	233	52	2,748.36	5.65	65.50	178	208	217	231	1058

Evaluation between Ann Arbor and João Pessoa

Figure 5.6 and Table 5.5 presents the results of RTT evaluation with all the services between ARB and JPA. The results of this evaluation presented similar values between Pusher and Unicast, which were unexpected. The possible reason is that the packets sent using Unicast were passing by São Paulo PoP instead of Fortaleza PoP, which is the closest to US. At the same time, packets sent using Pusher can follow routes outside the academic network before reaching the clusters.

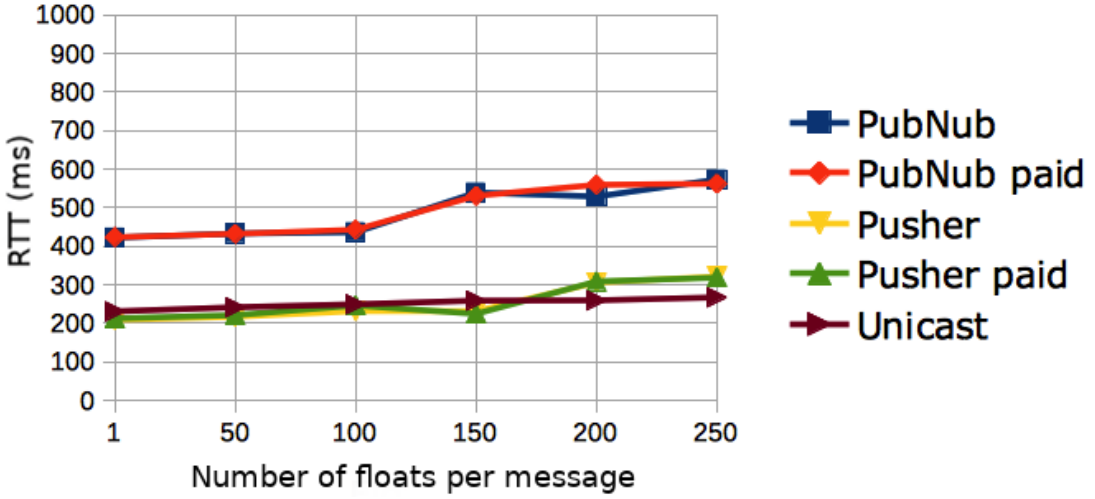


Figure 5.6: *Mean RTT between ARB and JPA for different services and message sizes. Delay between packages fixed in 150 ms*

Table 5.5: *Second experiment results: RTT evaluation between ARB and JPA*

Service	Size	Delay	Nobs	μ	σ	σ^2	Skew	Kurt	Min	q1	q2	q3	Max
pubnub	1	150	1000	480	298	89,045.38	4.92	26.81	364	383	394	420	2905
pubnub	50	150	1000	498	400	160,061.26	9.02	107.13	367	391	404	432	6737
pubnub	100	150	1000	493	447	199,905.79	10.21	123.85	373	405	418	439	7243
pubnub	150	150	1000	642	554	307,458.81	7.39	64.52	461	504	519	558	7237
pubnub	200	150	1000	661	597	356,749.09	5.39	31.07	461	488	513	549	5410
pubnub	250	150	1000	845	1863	3,475,934.57	9.46	95.36	471	522	548	609	**25559**
pubnubP	1	150	1000	497	437	191,571.06	8.31	88.06	357	382	391	421	6618
pubnubP	50	150	1000	486	351	123,638.71	8.82	98.78	366	389	400	428	5237
pubnubP	100	150	1000	566	834	696,442.12	10.98	142.63	374	406	420	452	13930
pubnubP	150	150	1000	573	370	136,770.67	12.47	179.77	462	503	516	541	7120
pubnubP	200	150	1000	666	401	160,645.05	3.76	15.15	463	511	530	599	3768
pubnubP	250	150	1000	635	293	86,179.32	3.79	16.20	471	517	535	583	2756
pusher	1	150	1000	**209**	32	1,012.05	4.24	27.14	**170**	192	205	215	503
pusher	50	150	1000	217	36	1,296.08	3.59	18.62	174	198	210	222	550
pusher	100	150	1000	233	58	3,347.10	6.20	64.38	172	206	217	242	1015
pusher	150	150	1000	234	77	5,973.63	6.04	44.86	177	203	218	232	1059
pusher	200	150	1000	306	46	2,103.95	1.10	5.66	201	279	303	334	654
pusher	250	150	**995**	336	157	24,688.01	7.57	73.07	190	273	320	356	2243
pusherP	1	150	1000	213	62	3,850.73	7.65	75.03	171	190	202	216	976
pusherP	50	150	1000	222	37	1,380.66	3.56	21.20	173	201	213	227	598
pusherP	100	150	**999**	250	95	9,124.09	5.80	51.87	177	206	220	256	1387
pusherP	150	150	1000	225	29	861.74	1.81	4.50	180	207	219	233	387
pusherP	200	150	**980**	396	783	613,390.44	8.89	77.74	194	278	307	339	8187
pusherP	250	150	1000	319	55	3,033.88	-0.01	-0.15	199	283	323	356	561
unicast	1	150	1000	231	15	230.51	9.71	106.79	222	228	229	231	427
unicast	50	150	1000	242	24	560.52	8.36	88.98	225	235	239	243	611
unicast	100	150	1000	250	35	1,252.13	11.92	183.23	226	241	245	251	915
unicast	150	150	1000	259	37	1,351.90	4.93	29.33	232	245	250	258	595
unicast	200	150	1000	260	41	1,713.40	10.96	146.77	234	248	254	263	973
unicast	250	150	1000	268	36	1,281.82	8.06	98.89	238	253	260	271	851

5.3 Third experiment: IPv4 versus IPv6

Tables 5.6 and 5.7 present the results of RTT evaluation with Unicast through IPv4 and IPv6 protocols between Ann Arbor and São Paulo. Figure 5.7 and 5.8 visually presents the results for RTT in IPv4 and IPv6, respectively, while Figure 5.9 presents the packet loss for IPv6 evaluation (observe that IPv4 test lost only one packet and this result is presented only in Table 5.6). This evaluation was conducted much later, on February, 2017, due to many technical reasons. We spent more than one year trying to setup Multicast between RNP and Internet2, but the service was unstable all the time. The network managers did their best, but in the end we decided to abort the Multicast evaluation due to time limit constraints.

In the end, the Unicast evaluation on IPv4 and IPv6 had interesting results. IPv6 seems to offer fastest paths than IPv4, although IPv4 presents less packet loss. The packet loss was present in all evaluations with IPv6 and the possible reason for this situation is that the protocol is still being implemented and evaluated at some paths of the academic network. Another possible reason is that some devices may work unpredictably with this protocol yet.

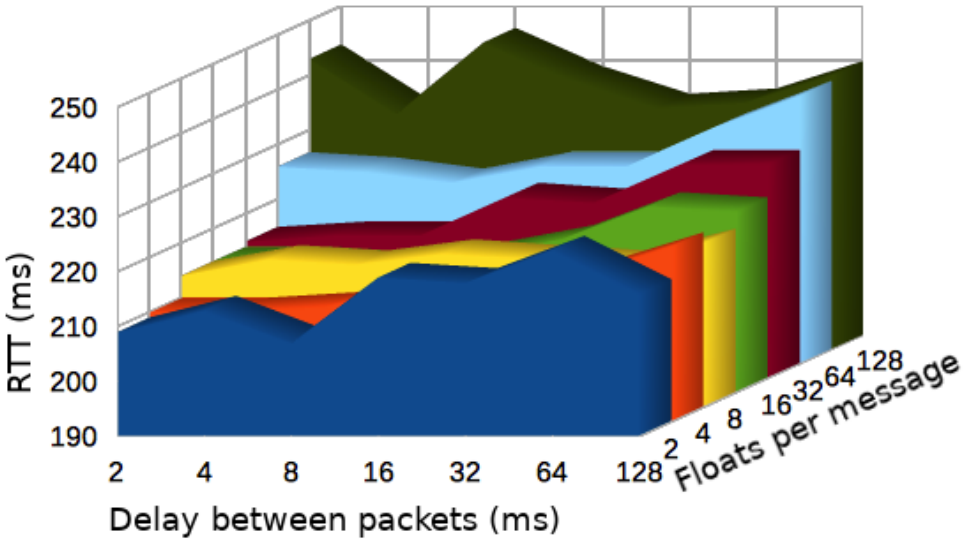


Figure 5.7: *Unicast IPv4: RTTs for different message sizes and delays between packets.*

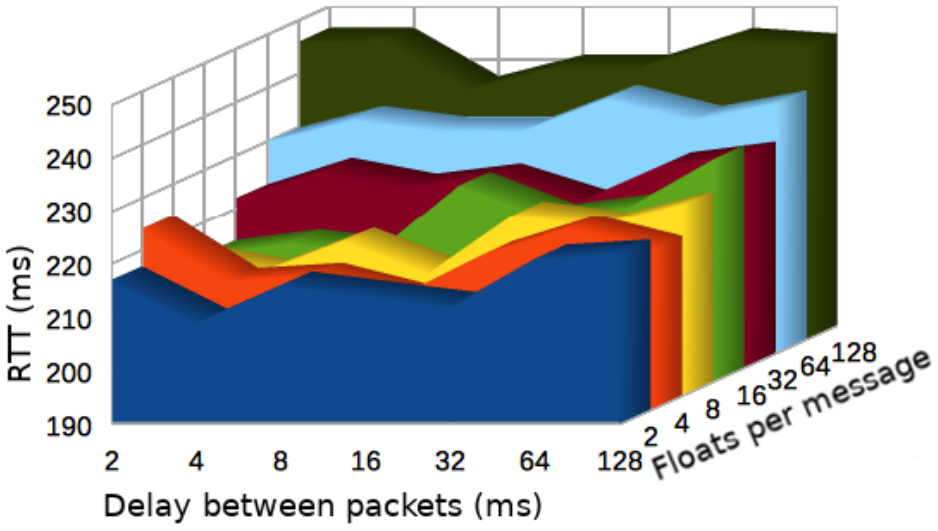


Figure 5.8: *Unicast IPv6: RTTs for different message sizes and delays between packets.*

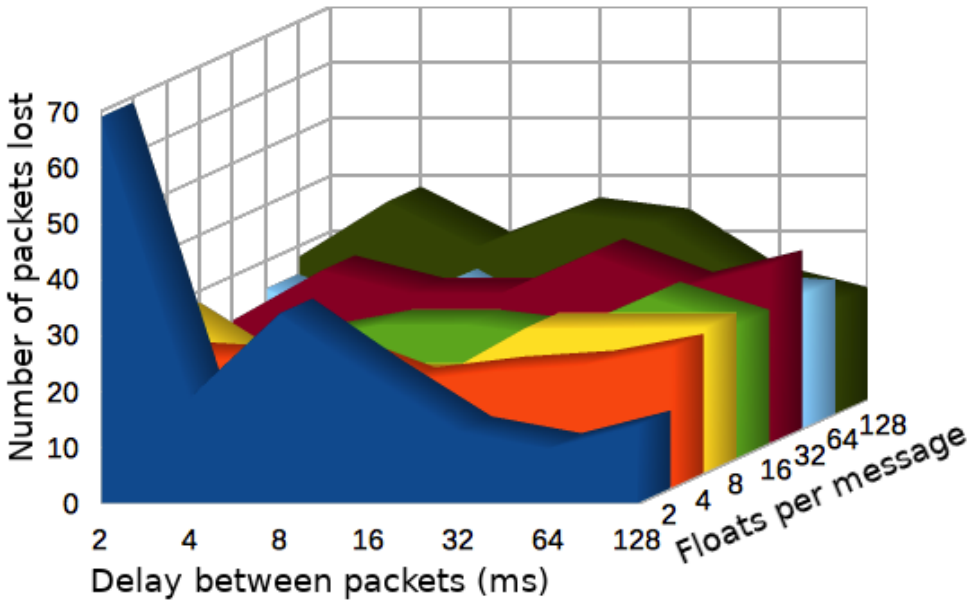


Figure 5.9: *Unicast IPv6: Packet loss for different message sizes and delays between packets.*

Table 5.6: *Third experiment results: IPv4 RTT evaluation between Ann Arbor and São Paulo*

Service	Size	Delay	Nobs	μ	σ	σ^2	Skew	Kurt	Min	q1	q2	q3	Max
unicastIPv4	2	2	1000	217	60	3,558.09	1.14	0.46	165	170	186	256	450
unicastIPv4	2	4	1000	**209**	53	2,793.07	1.28	0.85	**164**	170	175	243	412
unicastIPv4	2	8	1000	216	58	3,356.04	1.22	1.13	**164**	170	186	256	488
unicastIPv4	2	16	1000	214	55	3,013.90	1.14	0.42	**164**	171	183	252	417
unicastIPv4	2	32	1000	212	57	3,291.98	1.62	2.27	166	174	180	241	477
unicastIPv4	2	64	1000	221	56	3,133.15	1.31	1.07	167	181	190	252	431
unicastIPv4	2	128	1000	222	59	3,518.46	1.45	1.96	167	181	185	261	502
unicastIPv4	4	2	1000	224	60	3,632.01	1.04	0.63	**164**	171	205	265	491
unicastIPv4	4	4	**999**	214	57	3,300.74	1.21	0.68	**164**	170	180	251	443
unicastIPv4	4	8	1000	215	58	3,337.70	1.29	1.35	**164**	170	185	251	511
unicastIPv4	4	16	1000	211	56	3,111.82	1.51	1.94	165	171	178	243	503
unicastIPv4	4	32	1000	219	58	3,347.05	1.24	0.93	166	175	186	257	492
unicastIPv4	4	64	1000	224	59	3,524.73	1.48	2.03	167	182	193	257	507
unicastIPv4	4	128	1000	220	58	3,418.48	1.52	1.92	166	182	185	252	491
unicastIPv4	8	2	1000	216	61	3,668.89	1.31	1.00	**164**	171	178	251	485
unicastIPv4	8	4	1000	213	60	3,655.43	1.56	2.05	165	171	176	245	491
unicastIPv4	8	8	1000	219	61	3,707.13	1.32	1.45	166	171	184	258	513
unicastIPv4	8	16	1000	213	58	3,423.24	1.46	1.49	165	172	178	247	482
unicastIPv4	8	32	1000	224	60	3,641.28	1.29	1.38	166	176	197	259	493
unicastIPv4	8	64	1000	222	57	3,201.64	1.52	2.14	166	182	191	253	497
unicastIPv4	8	128	1000	226	63	3,992.91	1.63	2.79	169	183	187	263	535
unicastIPv4	16	2	1000	214	57	3,207.92	1.32	1.00	166	173	178	250	460
unicastIPv4	16	4	1000	216	55	3,031.27	1.14	0.39	165	173	182	255	420
unicastIPv4	16	8	1000	214	55	3,047.55	1.30	1.04	165	173	180	251	458
unicastIPv4	16	16	1000	227	63	4,021.29	1.18	0.79	166	175	198	265	489
unicastIPv4	16	32	1000	221	57	3,295.46	1.35	1.54	166	178	188	255	516
unicastIPv4	16	64	1000	223	54	2,879.77	1.37	1.31	168	185	192	254	456
unicastIPv4	16	128	1000	232	66	4,318.96	1.41	1.64	166	185	192	273	509
unicastIPv4	32	2	1000	222	58	3,392.14	1.17	0.71	167	177	188	264	469
unicastIPv4	32	4	1000	227	58	3,348.27	0.99	0.37	166	177	203	269	497
unicastIPv4	32	8	1000	224	62	3,873.42	1.40	1.71	166	177	189	262	497
unicastIPv4	32	16	1000	226	62	3,820.62	1.38	1.73	168	178	193	264	550
unicastIPv4	32	32	1000	221	57	3,221.42	1.50	2.08	166	181	190	254	518
unicastIPv4	32	64	1000	228	55	3,013.00	1.50	2.25	166	189	199	260	517
unicastIPv4	32	128	1000	230	61	3,728.85	1.55	2.32	170	188	194	267	511
unicastIPv4	64	2	1000	230	59	3,540.37	1.20	1.06	168	183	199	272	487
unicastIPv4	64	4	1000	234	58	3,386.02	0.94	0.13	168	184	215	275	478
unicastIPv4	64	8	1000	232	60	3,573.38	1.08	0.55	169	184	208	274	495
unicastIPv4	64	16	1000	232	59	3,513.60	1.30	1.41	171	186	200	271	506
unicastIPv4	64	32	1000	238	70	4,884.21	1.48	1.85	169	187	200	273	518
unicastIPv4	64	64	1000	234	53	2,818.47	1.26	0.84	172	196	207	264	454
unicastIPv4	64	128	1000	237	59	3,430.93	1.43	2.20	168	194	204	276	559
unicastIPv4	128	2	1000	246	68	4,618.75	1.49	3.63	172	192	220	293	**673**
unicastIPv4	128	4	1000	246	61	3,681.45	0.92	0.32	172	193	227	291	521
unicastIPv4	128	8	1000	237	61	3,677.11	1.32	1.42	174	192	206	276	511
unicastIPv4	128	16	1000	241	60	3,608.68	1.18	1.02	175	193	215	279	491
unicastIPv4	128	32	1000	241	59	3,498.73	1.31	1.57	174	195	213	279	537
unicastIPv4	128	64	1000	246	62	3,897.96	1.72	3.65	178	203	216	278	640
unicastIPv4	128	128	1000	245	64	4,084.91	1.59	2.39	179	201	213	278	527

Table 5.7: *Third experiment results: IPv6 RTT evaluation between Ann Arbor and São Paulo*

Service	Size	Delay	Nobs	μ	σ	σ^2	Skew	Kurt	Min	q1	q2	q3	Max
unicastIPv6	2	2	**931**	209	83	6,899.31	11.95	251.88	162	166	171	241	**2040**
unicastIPv6	2	4	**981**	213	61	3,780.19	1.38	1.60	**161**	167	176	250	512
unicastIPv6	2	8	**966**	**207**	56	3,142.12	1.35	1.18	**161**	167	172	243	470
unicastIPv6	2	16	**977**	219	59	3,508.90	1.04	0.27	**161**	168	194	259	460
unicastIPv6	2	32	**987**	218	56	3,165.50	1.16	1.02	162	172	194	255	505
unicastIPv6	2	64	**990**	224	61	3,664.21	1.35	1.53	162	178	198	257	487
unicastIPv6	2	128	**986**	216	56	3,093.35	1.54	2.21	164	178	184	248	508
unicastIPv6	4	2	**976**	210	60	3,637.20	1.47	1.75	**161**	167	172	244	502
unicastIPv6	4	4	**977**	210	58	3,361.79	1.37	1.39	162	167	173	246	495
unicastIPv6	4	8	**975**	211	57	3,247.00	1.24	0.84	162	167	177	248	447
unicastIPv6	4	16	**981**	214	61	3,684.01	1.45	1.88	**161**	168	179	247	493
unicastIPv6	4	32	**979**	213	57	3,248.35	1.35	1.09	162	172	179	247	459
unicastIPv6	4	64	**978**	217	58	3,323.11	1.73	3.37	164	178	186	246	506
unicastIPv6	4	128	**975**	222	64	4,120.21	1.63	2.48	164	179	183	256	498
unicastIPv6	8	2	**972**	214	60	3,565.14	1.29	1.35	162	168	179	252	513
unicastIPv6	8	4	**982**	217	59	3,525.46	1.17	0.73	**161**	168	187	255	470
unicastIPv6	8	8	**983**	216	68	4,617.50	1.62	2.39	162	168	174	249	506
unicastIPv6	8	16	**983**	218	62	3,812.69	1.39	1.81	162	170	190	254	506
unicastIPv6	8	32	**974**	217	60	3,643.81	1.48	1.97	162	173	182	251	502
unicastIPv6	8	64	**974**	216	53	2,818.20	1.42	1.50	164	180	186	247	440
unicastIPv6	8	128	**974**	220	57	3,205.65	1.36	1.32	164	180	185	254	496
unicastIPv6	16	2	**983**	214	56	3,186.99	1.17	0.46	162	170	180	248	417
unicastIPv6	16	4	**979**	214	58	3,402.94	1.23	0.71	162	170	178	253	453
unicastIPv6	16	8	**976**	213	56	3,094.19	1.24	0.73	163	171	179	249	419
unicastIPv6	16	16	**976**	215	60	3,547.88	1.45	1.81	163	171	181	251	502
unicastIPv6	16	32 with	**978**	218	60	3,577.64	1.50	1.99	164	175	183	251	496
unicastIPv6	16	64	**971**	224	58	3,334.72	1.34	1.37	166	181	194	258	478
unicastIPv6	16	128	**976**	223	60	3,634.23	1.61	2.68	166	182	187	259	517
unicastIPv6	32	2	**978**	215	56	3,137.36	1.30	1.10	165	174	182	253	481
unicastIPv6	32	4	**969**	216	58	3,374.93	1.29	0.79	164	174	181	252	459
unicastIPv6	32	8	**973**	216	58	3,338.28	1.30	1.07	163	173	181	253	476
unicastIPv6	32	16	**973**	223	59	3,505.97	1.23	1.08	164	176	193	261	482
unicastIPv6	32	32	**966**	222	57	3,278.71	1.28	1.05	165	179	190	257	470
unicastIPv6	32	64	**972**	229	57	3,201.44	1.33	1.38	169	187	200	262	497
unicastIPv6	32	128	**968**	229	58	3,425.28	1.31	1.14	168	186	194	264	486
unicastIPv6	64	2	**975**	226	60	3,619.86	1.21	0.94	165	180	191	265	483
unicastIPv6	64	4	**981**	225	59	3,535.05	1.24	1.06	166	180	191	265	487
unicastIPv6	64	8	**974**	223	60	3,615.61	1.45	1.88	166	180	189	259	515
unicastIPv6	64	16	**983**	226	59	3,542.56	1.36	1.68	167	182	194	265	514
unicastIPv6	64	32	**982**	226	58	3,396.10	1.61	2.96	166	184	195	261	523
unicastIPv6	64	64	**975**	233	59	3,536.28	1.69	3.14	167	191	204	262	516
unicastIPv6	64	128	**976**	239	64	4,159.53	1.60	3.15	173	192	205	277	621
unicastIPv6	128	2	**972**	243	66	4,323.65	1.10	0.85	169	188	216	288	535
unicastIPv6	128	4	**962**	233	59	3,471.90	1.18	0.81	170	188	201	273	488
unicastIPv6	128	8	**970**	246	79	6,213.39	1.46	1.69	169	187	202	286	576
unicastIPv6	128	16	**964**	239	59	3,519.94	1.08	0.50	171	192	209	280	474
unicastIPv6	128	32	**966**	234	54	2,971.64	1.18	0.65	171	193	206	270	444
unicastIPv6	128	64	**976**	235	51	2,578.85	1.37	1.40	173	200	211	263	481
unicastIPv6	128	128	**980**	240	59	3,486.10	1.52	2.27	175	199	209	275	531

Chapter 6

Conclusions

We are human
After all.
Flesh in common
After all.

Daft Punk

Mobile Music is a growing field that can still be considered in its initial stages of experimentation. Mobile technologies are improving and should continue to do so as mobile computing remains the focus of the market. Musicians may take advantage of these novelties within their compositional processes, and will benefit the most if they become aware of the constraints inherent to these technologies. This work thus proposed to evaluate these technologies and present diverse evaluations as well as applications as references for those who want to explore mobile music possibilities and experiment with appealing solutions for large scale user interaction, collaboration, and cooperation.

Delay between messages is an important parameter which was evaluated in the experiments here presented, with reasonable results that reveal the dependence of the statistics on the delay. Message loss may have occurred in our evaluations due to overhead on the network buffers: in the event that at least two packets are grouped by a network buffer before being sent, it is possible to have more than 10 messages per second even when using 150 ms delays between packets, which would be a problem for Cloud Services such as Pusher, but would work flawlessly for alternatives such as PubNub or Unicast.

When using Cloud Services it is possible that paid plans would offer different clusters that could be selected to optimize routes between performers and users of a mobile music application. Although clusters are available around the world for some Cloud Services, there are occasions when it may be necessary to try a different cloud service for better cluster localization and routing.

The cloud services used during this research had some specific settings depending on the plan. The cloud services free account was used most of the time but a paid plan was setup in order to compare the services during some evaluations. In this case, I had to use two different accounts and two pair of keys for each cloud service evaluated. The account set up and configuration were available through the user interface at the cloud service website. The free account is ready to use at the time of registering an account at the web site of each cloud service. After paying for the paid plan with the credit card, the new settings and limits become available automatically

Results obtained from all evaluations suggest that the mobile music community may use Cloud Services as an option for intercommunication in music applications, an alternative that facilitates implementation and improves scalabity, overcoming technological constraints faced by previous mobile music experiments restricted to small groups of participants. The delay and reliability of the service has proven to be suitable for present technologically-demanding applications, and we may expect improvements in a near future broadnening the range of possible applications. Push-notifications help users to send and receive data from any part of the world and control the final sound from their own mobile devices, using other users' inputs to influence their music creation in

a collaborative way. In this way, many musical works formerly conceived for local networks might now be extended to cloud-based services.

The evaluations using the academic network provided insights into many challenges and difficulties that may be faced by those who plan to use it. The initial configuration of the network and the requests for changing settings usually depend on many agents and may take from a few weeks to months or even years (in the case of IPv6 and Multicast). Although the throughput is much better in Unicast (and probably would be in Multicast) than using Cloud Services, the burden of the network setup may discourage all but the most tenacious mobile musicians. Network services such as IPv6 and Multicast were found to be still unstable within the academic network, despite their great theoretical advantages, and Multicast was untested due to years of recurring problems within network settings over the full network path, inbound and outbound.

It is important to notice that in order to use the academic network I had to contact many network managers from time to time. The RNP managers requested me a signed document assuming the responsibilities regarding any problem caused to the network during my evaluations. This document had to be signed by my advisor and they started managing the requested settings after receiving the document. The UMich network managers set-up everything considering my UMich number and email in order to register the requests under my responsibilities.

The settings requested during this research at both Universidade de São Paulo and University of Michigan was a VLAN with IPv4 and IPv6 blocks for interaction through Unicast and Multicast services, and also with Internet access for intercommunication with the Cloud Services. Although it seemed to be an easy request from my point of view, the services were unstable during most of the time of the research. The Multicast was the most troublesome setting and required many requests to Network Operations Center (NOC) throughout the whole path between Universidade de São Paulo and University of Michigan. The Unicast and Internet were available most of the time, on the other hand.

The applications developed during this research were heavily used and are still under constant development and improvement. Users around the world are still requesting new features, specially in the case of Sensors2PD, Sensors2OSC, SuperCopair, and Crowd in C[loud], which can be taken to mean that these applications have continued to attract some attention and interest even though their original performances and/or developing scenarios are long gone. Their code will remain open source in order to allow users to learn from these technologies and developers to collaborate and improve them, participating in the source code projects as members, which is an important aspect of the free open source software community.

6.1 Published papers: description

During the development of this research, many papers were published in order to help disseminate its results. A description of the international papers related to this thesis is given below.

The first paper published during this research process was "FFT benchmark on Android devices: Java versus JNI" (de Carvalho Junior *et al*, 2013), which discusses the effects using Java or JNI for DSP on Android devices. The results show that the programmer can use low level and parallel programming depending depending on hardware and OS. These options have better performance under specific circumstances, such as multi-core devices, newer OS, and larger block size. The full paper is presented in Appendix F.1.

The discussion presented in the paper named "Touches on the line: Sharing Csound scores using web server and mobile phones" (de Carvalho Junior, 2013) describes the possibilities of using a RoR web server in order to exchange notes or scores through the Internet. A web server created with RoR requires few commands and a single Ruby file. This proposal facilitates the development of interactive mobile performances for local and distributed network. The full paper is presented in Appendix F.2.

A discussion about new kind of performance without barriers and breaking with traditional art concepts is presented in the paper "Notes on the Elimination of the Mobile Music Audi-

ence" (Bandeira and de Carvalho Junior, 2014). It is inspired by the text "Notes on the Elimination of the Audience" by Allan Kaprow, and the paper also includes a description of the Hoketus application. The full paper is presented in Appendix F.3.

The first paper about Sensors2 applications was "Sensors2PD: Mobile sensors and WiFi information as input for Pure Data" (de Carvalho Junior, 2014), although the paper discussed in the previous paragraph used Sensors2PD initial code. This paper discuss the many possibilities of using Sensors2PD to interact with Pure Data patches on Android devices through this application. All sensors available in a device can act as an actuator and send data to receivers created on the patch. The full paper is presented in Appendix F.4.

"Indoor localization during installations using Wi-Fi" (de Carvalho Junior, 2015) presents the approach of using WiFi signal quality as an input for Sensors2PD during the installation "Hoketus". The discussion emphasizes the advantage of using WiFi instead of GPS for indoor localization at installations and performances. GPS signal is noisy or unavailable inside buildings with concrete structure and also has interference from the environment nature. On the other hand, WiFi signal can be disposed depending on user needs through many devices like routers, or smartphones hot-spots. The latter option was used during the installation. The full paper is presented in Appendix F.5.

"Sensors2OSC" (de Carvalho Junior and Mayer, 2015) discuss the approach to conceive this application and all its features. Sensors2OSC allows users to send samples from Android sensors through OSC inside a network. The application follows the same idea of Sensors2PD where the transmission of samples of a sensor is activated/deactivated in real-time, but includes the possibility of IP and network port definition for routing the se samples. The application and the paper came from a partnership with Thomas Mayer. The full paper is presented in Appendix F.6.

"Computer Music through the Cloud: Evaluating a Cloud Service for Collaborative Computer Music Applications" (de Carvalho Junior *et al*, 2015b) is a paper that summarize the first attempts to used Cloud Service for mobile music communication during this research. The paper shows results with data transmission through Pusher Cloud Service between Brazil and USA. The evaluation considered data with different sizes and gave insights to the advantage of using this service at an unusual situation. The full paper is presented in Appendix F.7.

The Pusher Cloud Service is also evaluated at the paper "SuperCopair: Collaborative Live Coding on SuperCollider through the cloud" (de Carvalho Junior *et al*, 2015a). In this case, the service is evaluated under unusual circumstances by means of an application for pair-programming live coding. SuperCopair application allows users to collaborate inside a live coding session using the SuperCollider language, share the code, and share the code synthesis through the Internet. The burden to initiate a collaborative live coding session is surpassed by one or two shortcuts inside Atom.io IDE. The sound synthesis happens inside the IDE while the code is shared through the Cloud Service. The full paper is presented in Appendix F.8.

"Cooperative Live Coding as an instructional model" (de Carvalho Junior, 2015) discuss the experiences of using SuperCopair with users from different cities and countries in real-time. The idea of cooperation is highlighted because the users helped themselves fixing codes and writing comments with suggestions close to friends code. during the live sessions. The full paper is presented in Appendix F.9.

"Crowd in C[loud]: Audience Participation Music with Online Dating Metaphor using Cloud Service" (Lee *et al*, 2016) is paper about an experience with Cloud Services in Mobile Music. The piece "Crowd in C[loud]" took advantage of Web Audio technology for sound synthesis through the browser of mobile phones and allowed an online dating of melodies using Cloud Services for network interaction in real-time during a live performance inside a theater. Around 60 participants joined the session which last for 10 minutes. An important point about this piece is that the audio came only from participants devices but could be heard by the whole audience. The full paper is presented in Appendix F.10.

A discussion about the network communication is presented in "Understanding Cloud Service in the Audience Music Performance of Crowd in C[loud]" (de Carvalho Junior *et al*, 2016). During the performance we had a backup computer connected to the Cloud Service monitoring the net-

work communication in all channels created on the Cloud Service. Data consumption and traffic is presented as motivation for using Cloud technologies in performances due to its capacity and scalability. The full paper is presented in Appendix F.11.

"Open band: Audience Creative Participation Using Audio Synthesis" (Stolfi *et al*, 2017b) is another performance/installation using Web Audio for sound synthesis including audience interaction through the network. The communication can occur within local network or Cloud Services, depending on the proposal. This paper discuss the technical aspects and advantages of using Web Audio synthesis for this performance. The full paper is presented in Appendix F.12.

"Open Band: A Platform for Collective Sound Dialogues" (Stolfi *et al*, 2017a) discuss Open Band performance in terms of the user experience during the performances. Local and distributed performances favored distinct evaluations regarding the same application. The full paper is presented in Appendix F.13.

dj@ime.usp.br

Appendix A

Evolution of smartphone features

Regarding the upgrades throughout the years we can see here that in 10 years the processors are almost 10 times faster in each core, the memory capacity is 25 times bigger now, and the communication technologies increased the number of network bands and the bandwidth. It is interesting to notice that in 2006 we already have a smartphone with WLAN, 3G, Bluetooth, infrared, GPS and radio technologies into a single board.

Table A.1: *Communication technologies available on smartphones with 10 years announcement difference: Nokia N95 (2006) and Samsung Galaxy S7 (2016). Source: Adapted from GSMArena (2016)*

		Nokia N95	Samsung Galaxy S7
LAUNCH	Announced	2006, September. Released 2007, March	2016, February
	Status	Discontinued	Available. Released 2016, March
PLATFORM	CPU	332 MHz Dual ARM 11	Dual-core 2.15 GHz Kryo & Dual-core 1.6 GHz Kryo or Quad-core 2.3 GHz Mongoose + Quad-core 1.6 GHz Cortex-A53
MEMORY	Card slot	microSD, up to 8 GB (dedicated slot), 128 MB included	microSD, up to 200 GB (dedicated slot) microSD, up to 200 GB
	Internal	160 MB, 64 MB RAM	32/64 GB, 4 GB RAM
NETWORK	Technology	GSM / HSPA	GSM / HSPA / LTE
	2G bands	GSM 850 / 900 / 1800 / 1900	GSM 850 / 900 / 1800 / 1900
	3G Network	HSDPA 2100	HSDPA 850 / 900 / 1700(AWS) / 1900 / 2100 - G930F
		HSDPA 850 / 1900	TD-SCDMA
	4G Network		LTE band 1(2100), 2(1900), 3(1800), 4(1700/2100), 5(850), 7(2600), 8(900), 12(700), 13(700), 17(700), 18(800), 19(800), 20(800), 25(1900), 26(850), 28(700), 38(2600), 39(1900), 40(2300), 41(2500) - G930F
	Speed	HSPA	HSPA 42.2/5.76 Mbps, LTE Cat9 450/50 Mbps
	GPRS	Class 10	Yes

Table A.1: *Communication technologies available on smartphones with 10 years announcement difference: Nokia N95 (2006) and Samsung Galaxy S7 (2016). Source: Adapted from GSMArena (2016)*

		Nokia N95	Samsung Galaxy S7
	EDGE	Class 32, 296 kbps; DTM Class 11, 177 kbps	Yes
COMMS	WLAN	Wi-Fi 802.11 b/g, UPnP technology	Wi-Fi 802.11 a/b/g/n/ac, dual-band, Wi-Fi Direct, hotspot
	Bluetooth	v2.0, A2DP	v4.2, A2DP, LE, aptX
	GPS	Yes, with A-GPS; Nokia Maps	Yes, with A-GPS, GLONASS, BDS
	NFC	No	Yes
	Infrared port	Yes	No
	Radio	Stereo FM radio	No
	USB	miniUSB v2.0	microUSB v2.0, USB Host

Appendix B

Available sensors within the Android system

This is a list of some available sensors for mobile devices, being the base sensors the ones that are related to physical sensors while the others are composite sensors that use values from the base sensors or merge two or more sensors.

Table B.1: *Definition of available sensors for Android system. Source: The Android Open Source Project (2016a,b)*

Sensor type	ID	Reporting mode	# of values	Composite category	Low power
ACCELEROMETER	1	continuous	3	(base sensor)	
GEOMAGNETIC_FIELD	2	continuous	3	(base sensor)	
ORIENTATION	3	continuous	3	attitude	
GYROSCOPE	4	continuous	3	(base sensor)	
LIGHT	5	on-change	1	(base sensor)	
PRESSURE	6	continuous	1	(base sensor)	
TEMPERATURE	7		1		
PROXIMITY	8	on-change	1	(base sensor)	
GRAVITY	9	continuous	3	attitude	
LINEAR_ACCELERATION	10	continuous	3	activity	
ROTATION_VECTOR	11	continuous	4	attitude	
RELATIVE_HUMIDITY	12	on-change	1	(base sensor)	
AMBIENT_TEMPERATURE	13	on-change	1	(base sensor)	
MAGNETIC_FIELD_UNCALIBRATED	14	continuous	6	uncalibrated	
GAME_ROTATION_VECTOR	15	continuous	3	attitude	
GYROSCOPE_UNCALIBRATED	16	continuous	6	uncalibrated	
SIGNIFICANT_MOTION	17	one-shot	1	activity	yes
STEP_DETECTOR	18	special	1	activity	yes
STEP_COUNTER	19	on-change	1	activity	yes
GEOMAGNETIC_ROTATION_VECTOR	20	continuous	4	attitude	yes
HEART_RATE	21	on-change	1	(base sensor)	
TILT_DETECTOR	22	special	1	activity	yes
WAKE_GESTURE	23	one-shot	1	interaction	yes
GLANCE_GESTURE	24	one-shot	1	interaction	yes
PICK_UP_GESTURE	25	one-shot	1	interaction	yes
WRIST_TILT_GESTURE	26	special	1		yes

Appendix C

Devices used during research

The mobile devices used during this research were the LG D686 G Pro Dual Lite, Samsung GT-I9300 Galaxy SIII, and Sony Xperia D8533 (and D8503) Z3 Compact. Their technical specifications are presented on the Table C.1.

Table C.1: *Technical specifications of the mobile devices used during this research: D686, S3, and Z3. Source: Adapted from GSMArena (2017)*

		LG D686 G Pro Lite Dual	Samsung GT-I9300 Galaxy SIII	Sony D8533 (and D8503) Xperia Z3 Compact
NETWORK	Technology	GSM / HSPA	GSM / HSPA	GSM / HSPA / LTE
	2G bands	GSM 850 / 900 / 1800 / 1900 - SIM 1 & SIM 2	GSM 850 / 900 / 1800 / 1900	GSM 850 / 900 / 1800 / 1900
	3G Network	HSDPA 850 / 900 / 1900 / 2100	HSDPA 850 / 900 / 1900 / 2100	HSDPA 850 / 900 / 1700 / 1900 / 2100 - D5803 HSDPA 850 / 900 / 1900 / 2100 - D5833
	4G Network			LTE band 1(2100), 2(1900), 3(1800), 4(1700/2100), 5(850), 7(2600), 8(900), 13(700), 17(700), 20(800) - D5803 LTE band 1(2100), 3(1800), 5(850), 7(2600), 8(900), 28(700), 40(2300) - D5833
	Speed	HSPA 7.2/5.76 Mbps	HSPA 21.1/5.76 Mbps	HSPA 42.2/5.76 Mbps, LTE Cat4 150/50 Mbps
	GPRS	Class 12	Class 12	Up to 107 kbps
	EDGE	Class 12	Class 12	Up to 296 kbps
LAUNCH	Announced	2013, October	2012, May	2014, September
	Status	Available. Released 2013, November	Available. Released 2012, May	Available. Released 2014, September

Table C.1: *Technical specifications of the mobile devices used during this research: D686, S3, and Z3. Source: Adapted from GSMArena (2017)*

		LG D686 G Pro Lite Dual	Samsung GT-I9300 Galaxy SIII	Sony D8533 (and D8503) Xperia Z3 Compact
PLATFORM	OS	Android OS, v4.1.2 (Jelly Bean)	Android OS, v4.0.4 (Ice Cream Sandwich), 4.3 (Jelly Bean)	Android OS, v4.4.4 (KitKat), upgradable to v6.0 (Marshmallow)
	Chipset	Mediatek MT6577	Exynos 4412 Quad	Qualcomm MSM8974AC Snapdragon 801
	CPU	Dual-core 1.0 GHz Cortex-A9	Quad-core 1.4 GHz Cortex-A9	Quad-core 2.5 GHz Krait 400
	GPU	PowerVR SGX531	Mali-400MP4	Adreno 330
MEMORY	Card slot	microSD, up to 32 GB (dedicated slot)	microSD, up to 64 GB (dedicated slot)	microSD, up to 256 GB (dedicated slot)
	Internal	8 GB, 1 GB RAM	16/32/64 GB, 1 GB RAM	16 GB, 2 GB RAM
COMMS	WLAN	Wi-Fi 802.11 b/g/n, Wi-Fi Direct, DLNA, hotspot	Wi-Fi 802.11 a/b/g/n, dual-band, Wi-Fi Direct, DLNA, hotspot	Wi-Fi 802.11 a/b/g/n/ac, dual-band, Wi-Fi Direct, DLNA, hotspot
	Bluetooth	v3.0, A2DP	v4.0, A2DP, EDR, aptX	v4.0, A2DP, LE, aptX
	GPS	Yes, with A-GPS	Yes, with A-GPS, GLONASS	Yes, with A-GPS, GLONASS
	NFC		Yes	Yes
	Infrared port	Yes	No	No
	Radio	FM radio	Stereo FM radio, RDS	Stereo FM radio, RDS
	USB	microUSB v2.0	microUSB v2.0 (MHL TV-out), USB Host	microUSB v2.0 (MHL TV-out), USB Host; magnetic connector
FEATURES	Sensors	Accelerometer, proximity, compass	Accelerometer, gyro, proximity, compass, barometer	Accelerometer, gyro, proximity, compass, barometer
BATTERY		Removable Li-Ion 3140 mAh battery	Removable Li-Ion 2100 mAh battery	Non-removable Li-Ion 2600 mAh battery
	Stand-by	Up to 845 h	Up to 590 h (2G) / Up to 790 h (3G)	Up to 880 h (2G) / Up to 920 h (3G)
	Talk time	Up to 14 h 30 min	Up to 21 h 40 min (2G) / Up to 11 h 40 min (3G)	Up to 12 h (2G) / Up to 14 h (3G)
	Music play			Up to 110 h

Two routers were bought in order to be used during the evaluation. One router was maintained at USP while the other was situated at UMich. The technical details about the routers are described

at Table C.2.

Table C.2: *Technical specifications of the TP-Link AC1750 Archer C7 Wireless Dual Band Gigabit Router. Source: Addapted from LINK (2017)*

HARDWARE FEATURES	
Interface	4 10/100/1000Mbps LAN Ports
	1 10/100/1000Mbps WAN Port
	2 USB 2.0 Ports
Button	WPS/Reset Button
	Wireless On/Off Switch
	Power On/Off Button
External Power Supply	12VDC / 2A
Dimensions (W x D x H)	9.6x6.4x1.3 in. (243x160.6x32.5mm)
Antenna Type	Three detachable antennas (RP-SMA)
WIRELESS FEATURES	
Wireless Standards	IEEE 802.11ac/n/a 5GHz
	IEEE 802.11b/g/n 2.4GHz
Frequency	2.4GHz and 5GHz
Signal Rate	5GHz: Up to 1300Mbps
	2.4GHz: Up to 450Mbps
Transmit Power	CE:
	<20dBm(2.4GHz)
	<23dBm(5GHz)
	FCC:
	<30dBm
Wireless Functions	Enable/Disable Wireless Radio, WDS Bridge, WMM, Wireless Statistics
Wireless Security	64/128-bit WEP, WPA / WPA2, WPA-PSK/ WPA2-PSK encryption
SOFTWARE FEATURES	
Quality of Service	WMM, Bandwidth Control
WAN Type	Dynamic IP/Static IP/PPPoE/PPTP(Dual Access)/L2TP(Dual Access)/BigPond
Management	Access Control
	Local Management
	Remote Management
DHCP	Server, Client, DHCP Client List, Address Reservation
Port Forwarding	Virtual Server, Port Triggering, UPnP, DMZ
Dynamic DNS	DynDns, Comexe, NO-IP
VPN Pass-Through	PPTP, L2TP, IPSec
Access Control	Parental Control, Local Management Control, Host List, Access Schedule, Rule Management
Firewall Security	DoS, SPI Firewall
	IP Address Filter/MAC Address Filter/Domain Filter
	IP and MAC Address Binding
Protocols	Supports IPv4 and IPv6
USB Sharing	Support Samba(Storage)/FTP Server/Media Server/Printer Server
Guest Network	2.4GHz guest network x 1
	5GHz guest network x 1
OTHERS	
Certification	CE, FCC, RoHS

Appendix D

Partnerships

Partnerships were established as a way to leverage research and profit from external contributions, while at the same time applying in practical situations theoretical work developed for the sake of exploring possibilities in mobile music. Some partners were selected based on a perceived synergy between their projects and this research, while others appeared after discussions and proposals based on ongoing research. In the following, a few of the interacting research groups and projects which contributed to the methodology adopted are briefly described.

Network configuration and evaluation

The main partnerships for the technical part of this research came from network administrators and laboratory members. In order to setup a complete route between many universities around the Americas, many people were contacted to set up configurations at the endpoints and in the middle of the routes as well. We had devices at University of São Paulo, Universidade Federal da Paraíba, and University of Michigan. The routes between these universities crossed the academic networks managed by RNP, RedeCLARA, and Internet2.

For the communication setup through the RNP network, system analyst Valter Pereira from POP-USP was the main contact and support from the beginning to the end of this research. At USP we also had assistance from Luiz Eduardo Silva dos Santos, Andre Lopes da Silva, and also the SI team from IME to setup the network communication from the Compmus laboratory to the POP-USP. Long distance communication with mobile devices situated at the Federal University of Paraíba was made possible due to the contact with the Carlos André Lacerda de Carvalho and Victor Igor de Lima Andrade, members of the Digital Video Applications Lab (LAViD)[1] at the time of the research.

The network setup at University of Michigan had support from Kyle Banas, Roy Hockett, and Brady Farver. Brian Pullin and Nathan Miller from Global NOC of Internet2 helped with the connection on the borders of USA. Christian O'Flaherty, Iara Machado, and Marco Teixeira were responsible for the final setup at RedCLARA, between Brazil and USA.

Computer Music Research Group

This research was conducted within the Computer Music (CompMus) Research Group at IME-USP, which provided many opportunities for partnerships and exchanging ideas. Regular weekly discussions allowed the dissemination of particular research problems by group members and also provided a space for research interactions and intersections.

A project focused on multichannel audio transmission over computer networks was led by Flávio Luiz Schiavoni and overlapped with the beginning of this research. His project's questions provided many opportunities for discussions regarding the many ways of dealing with data transmission over network protocols. Many insights and alternatives for music collaboration came from the experiments and performances using the Medusa framework created by Flávio.

[1] LAViD website: http://lavid.ufpb.br/ (visited on December 2018)

A project focused on realtime audio processing in mobile devices and developed by André Jucovsky Bianchi provided many essential interchanges of ideas during the preliminary/defining stages of this research. Setting up a partnership with Bianchi was a natural move based on my initial interests with Android application development and audio processing evaluation. The result of this partnership was the contribution in the development of the application *DSPBenchmarking* and of some algorithms evaluated through this tool, and is described in Appendix E.2.

Many evaluations here presented had to be coordinated between places and groups far away from each other, requiring the recruitment of volunteers to participate in the setups, who besides helping to conduct the evaluations, also made contributions to this research. Gilmar Rocha de Oliveira Dias, Thilo Koch, and Guilherme Feulo do Espírito Santo participated actively during most of the experiments with the *PushLoop* application described in Section 4.3.

Another important partnership formed towards the final stages of this research was with Fábio Goródscy, whose research interests started with the exploration of the Web Audio API. He had developed the *Open Band* project in collaboration with Ariane Stolfi (from the NuSom group discussed below) for audience interaction based on sampling, and we decided to build together a new lightweight version using only the Web Audio API, which is presented in Appendix E.10.

NuSom – Research Center on Sonology

The Research Center on Sonology (NuSom) is an interdepartmental research group that congregates the Computer Music Research Group at IME-USP and the Sonology research group at the School of Communication and Arts (ECA-USP), formed by artists-researchers interested in sound studies, composition, performance and improvisation and their relationships with technology. The partnership between these groups is strengthened by regular meetings, in which the members discuss text, projects, and performances.

At the beginning of this project, members of NuSom were involved in network music concerts connecting musicians at USP with other centers such as SARC/QUB (Ireland), CCRMA/Stanford (USA) and IRCAM (France). Even though no partnerships were formed at that time, their work served as motivation for considering distributed performance scenarios, and their approaches and solutions contributed to the development of this research.

The most important partner from NuSom was André Damião Bandeira, which is a contemporary composer and researcher studying mobile music and aiming to use some new technologies for his projects and performances. A collaboration emerged that led us to the development of *Hoketus* presented in Appendix E.5, which merged many technological and musical concepts into a musical distributed installation.

Residuum

The Pure Data discussion list is the main place for posting questions and solutions for Pure Data users. The collaboration with Thomas Mayer from Residuum[2] took place after his post on that list regarding the idea of using Android sensors to control Pure Data patches. At that time, *Sensors2Pd* was already published and running on the *Hoketus* installation. This interaction through the list offered us the opportunity to start a collaboration within a larger project named Sensors2, described in Appendix E.4.

The collaboration with Thomas resulted in the development of *Sensors2OSC* application, an improvement of *Sensors2Pd*, and many other ideas for applications. This project is receiving contributions from other developers around the world and the current applications are translated into four languages (English, German, Portuguese and Japanese).

[2]Residuum web site: https://www.residuum.org/ (visited on May, 2018)

University of Michigan

A period as an exchange student at the University of Michigan was planned during this research project, in order to develop partnerships and also to create opportunities for evaluating long distance communications. The research group at the University of Michigan, headed by Professor Georg Essl, was a natural choice due to their experience with mobile music solutions and their research involving applications for mobile platforms such as Android and iOS devices. Other members of the group were also invaluable partners, such as Sang Won Lee, Qi Yang, and Biqiao (Didi) Zhang.

An important aspect of the collaboration with professor Essl was the opportunity of working with the *urMus* platform, developed by him. This platform was designed as a complete mobile solution for mobile music experiences that works both in Android and iOS devices. The development of applications for *urMus* is made using the Lua language through a web interface, and many of these applications have been created by students from many different fields, such as computer science, engineering, and arts. The fact that the *urMus* platform can be programmed by students with such diverse backgrounds became a focus of interest for this research.

A local network at the University of Michigan was used during the research process with iOS devices. Application development with the *urMus* platform was mostly based on the knowledge acquired while attending the multidisciplinary course "Building a Mobile Phone Ensemble", taught by Professor Essl in 2015. In this course, many exploratory exercises were proposed, freeing students' creativity leading to many interesting/unpredictable applications. The idea of exploring local networks within exercises was an opportunity to evaluate mobile music interaction strategies in this context, with first attempts related to Unicast, Multicast, and the Bonjour technology. These experiences are described in Appendix E.9.

Appendix E

Applications created during the research

Many of the applications presented here are intentionally similar to other applications described in Chapter 2. At times the idea was to simplify codes and settings for novices, such as in *theremini-nimal* (Section E.1) and *pubslides* (Section E.6). However, some applications were created to offer solutions for gaps and needs identified in the literature, such as *Sensors2Pd* (Section E.4) and *Sensors2OSC* (Section E.4). Application's open source code exemplify the use of the technologies and allow other researchers to get through code examples for getting started with MM. The topics below will be presented following the conception order of the applications and the personal learning order of the technologies.

Partnership with musicians, researchers, and developers was a methodological approach for aggregating knowledge and experience from the community of potentially interested parties to this research. Many of the performances were only possible due to the musical conception of fellow musicians and artist-researchers. Some applications received contributions from the developers' community and also from artist-researchers that collaborated with their development from the initial conception to the evaluation process.

The initial motivation for these applications was indeed two courses taken during the first semester of 2012. The first was the course of "*Computação Móvel*" (Mobile Computing). The professor at that time was Alfredo Goldman vel Lejbman and the Android teaching assistant was Andrew Toshiaki Nakayama Kurauchi. During the course I learned the characteristics that makes Mobile Computing different from regular computing, such as the attention with the network communication, the battery consumption, and the mobility advantages. I had to develop an Android application and work with delay-tolerant networking (DTN) during the discipline under the supervision of the professor and his assistant. The classes and projects made me curious regarding the mobile technology and encouraged me to learn continuously about Mobile Computing.

The second one was a Pure Data course ministered at "*Museu da Imagem e do Som de São Paulo* " (São Paulo Museum of Image and Sound), known as MIS in São Paulo, SP, Brazil. The professor was Alexandre Porres Torres, a member of Compmus, which was an organizer of PDcon in 2009 that happened in Brazil. I learned the basics of Pure Data during this course and also got enchanted with the language and its possibilities. Another point that made me interested in studying the language was the learning curve and their popularity in other fields like Music, Architecture, and Design.

After taking these two courses during the first semester of 2012, I got in contact with the book Making Musical Apps (Brinkmann, 2012). Using this book I learned how to create my first mobile application using Pure Data and Android, which is the thereminimal presented at Section E.1. From this time on, I created new applications based on personal studies, experiments, and partnerships. These applications are described in this chapter.

E.1 Android and Pure Data: thereminimal

The first experiments with MM were related to user interaction through Android sensors. These experiments resulted in a simple application inspired by the Theremin instrument which is a classical

electrical instrument that presents two antennas connected to a circuit that controls audio oscillators based on the magnetic field interference caused by musicians hands (Glinsky, 1992).

The application *thereminimal* is a simple application with two main files proposing the control of an oscillator with two sensors. One sensor is selected to control the frequency and the other sensor controls the amplitude of the oscillator. The sound synthesis used for this application was based on a Pure Data patch running with the libpd library. The patch used in this application is presented in Figure E.1.

The application sends sensors events to the Pd receivers and these values are used to control the patch variables. The values of the sensors are converted before sending to the patch depending on each sensor. A range of values for minimum and maximum pitches is defined as 200 and 2000 Hz, respectively, and the volume range is from zero to one. The sensor maximum and minimum values are used in this conversion based on settings defined at the API, but these minimum and maximum ranges are also updated during the audio processing in case of divergence. UML Class Diagram of thereminimal is presented in Figure E.2 and screenshots of the application are presented in Figure E.3.

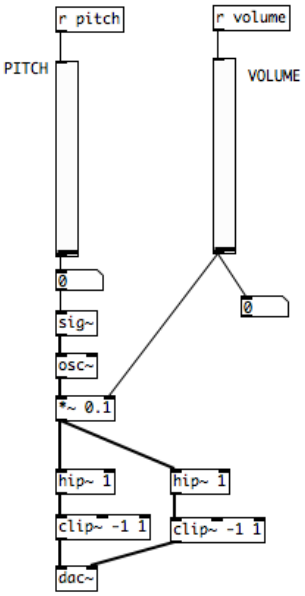


Figure E.1: *Pure Data patch used in thereminimal application.*

The first experiences with this application provided helped on learning about the Android sensor events. Basically, every event is captured by the API and is sent to the applications that are listening to these events depending on the predefined sample rate at application side. The fastest sample rate option was selected for thereminimal application; this means that as soon as an event is generated, it is received by the application.

The variations in the values are too high that the changes in the frequencies are perceived as single separated notes instead of ramps or glissando. On the other hand, the changes in the amplitude are imperceptible by most of users.

The code of the application is distributed with open source on github at the address: https://github.com/deusanyjunior/thereminimal (visited on May, 2017). A discussion about using libpd with Android applications was presented during a talk in the regular Compmus seminars at IME-USP on October 18, 2012[1].

[1]Talk about libpd and Android: http://compmus.ime.usp.br/en/node/320 (visited on May, 2017)

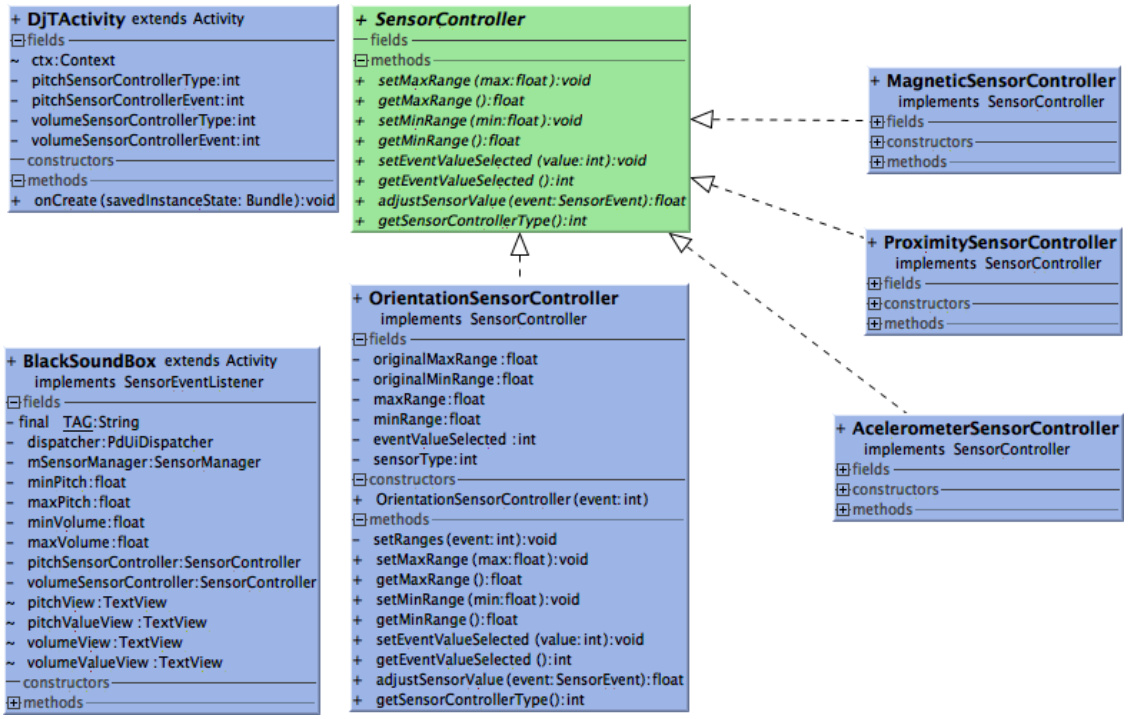
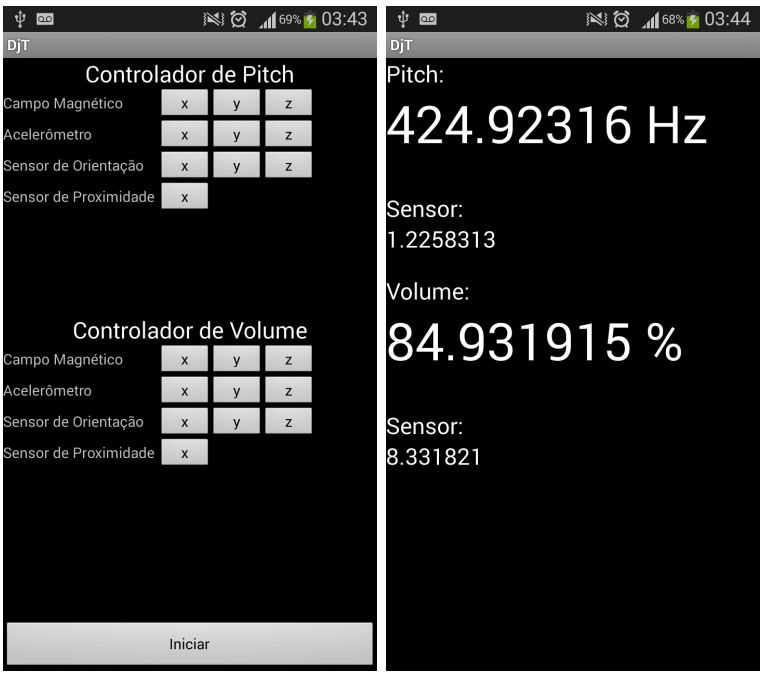


Figure E.2: *UML Class Diagram of thereminimal application.*



(a) *First screen* **(b)** *Second screen*

Figure E.3: *Screen-shots of thereminimal application.*

E.2 Java and JNI FFT on Android: Contribution to DSPBenchmarking

The evaluation of audio processing on Android devices was part of the Master project conducted by the researcher Andre Bianchi at the Compmus while I was initiating my PhD research. As soon as I learned the basics of Android application development, I started to collaborate with this project together with the researcher Max Rosan, that attended the Mobile Computing class with me.

The project evaluated many DSP methods in many Android devices focusing on the performance limits depending on the audio block size and the time required for finishing the process. Based on the first results of this evaluation described in Bianchi and Queiroz (2012), new methods were added in order to verify the performance of some FFT variations: traditional FFT, FFTW, multi-thread FFTW, double FFT, multi-thread double FFT. While the FFTW is using JNI for calculating the FFTW coded in C language, the double FFT is based on the JTransforms library for parallel FFT calculation in Java language.

Although better performance was expected with these non-traditional approaches, running C/C++ code for small blocks increases the complexity of the application and also the cost for processing the same audio block. A discussion regarding this evaluation was published as a paper at the 14th Brazilian Symposium of Computer Music (SBCM) (de Carvalho Junior *et al*, 2013) and is presented in Appendix F.1.

The multi-thread FFT had worst results than the traditional FFT, and some reasons are discussed below. The threads were created using native thread methods available for Java language, these threads probably shared the same core of application, and in the end the threads ran concurrently. An explanation for this behavior is that Android programming methodologies do require the use of special classes such as AsynkTask, or the packets Executor, ThreadPoolExecutor, and FutureTask. Depending on the API and the device limitations, the threads created with these classes will provide better performance, but this information was unknown at the time of the evaluation. Some other reasons are related to Android constraints regarding the parallelism, as the operating system decides about the threads distribution and even the activation of multi-cores. In this case, we found out that concurrency in Android applications is a topic that requires deep discussion and turns out to be outside of the scope of this research.

The code of the application is distributed with open source on github at the address: https://github.com/andrejb/DspBenchmarking (visited on May, 2017).

E.3 Android, CSound, and RoR: Touches On The Line

The first attempt to create collaborative applications for Mobile Music was through the use of a web service programmed using Ruby on Rails. The idea of creating a web service was the first idea for this solution considering my personal background as web developer. The technologies selected were based on the recent ones learned at this time.

During the period that I was Teaching Assitant for Mobile Computing course, I had to create many web services for the assignments proposed to the students and I had to choose a platform available for rapid application development (RAD). The Ruby on Rails (RoR) was one of the available options at that time which caught the attention due to its simplistic way to create online solutions. Another advantage for this solution is that the POST and GET requests using JSON format were added to the web service with few lines of code.

The CSound portability for Android devices (Yi and Lazzarini, 2012) guided the use of CSound for mobile applications because it presented many examples using numerous approaches for interaction through Android inputs and CSound scores. The *MultiTouchXY*[2] application was selected as a starting point for the development of *Touches On The Line* application due to the high sample rate of events available from touch events. Touch responsiveness on Android devices can have values up to 120 Hz and provides up to 10 fingers position, what implies 10 pair of position values (x and y) on every 8 ms in the best situation (Padre, 2017).

Touches On The Line application was defined to share touch's movements online and have its path synthesized as sound. The *MultiTouchXY* demo application was provided with a CSound score that receives two values and control a sound synthesis varying frequency and timber[3]. The demo

[2] MultiTouchXY code: https://github.com/csound/csound/blob/develop/android/CsoundAndroidExamples/src/com/csounds/examples/tests/MultiTouchXYActivity.java (visited on May, 2017)

[3] MultiTouchXY score for CSound: https://github.com/csound/csound/blob/develop/android/CsoundAndroidExamples/res/raw/multitouch_xy.csd (visited on May, 2017)

was first changed to send every new event to the web server, and to request new available values from the server all the time. This approach was changed even during the development due to many reasons: the communication with the web server was taking too long; the application was getting unresponsive when too many single touch events were sent one after another, due to many processes regarding post requisitions running in background; and the reception of too many single data (one by one) was taking too long.

The latest version of this application was developed with a buffer for a touch movement from the time the finger touches the screen to the release, including the delay between each timestamp received with the finger event. The finger movement is synthesized in audio inside the local device by sending every single movement coordinate to the CSound synthesizer as an argument for the synthesis The complete movement path is then sent to the web service after releasing the finger from the screen. This data is exchanged with the web service using JSON format and contains only float numbers for position and integer numbers for delays in milliseconds.

Although this application allows many devices to participate in the same session, it was evaluated only with three devices available at lab during the period of its conception. After its development, the application was used in some solo performances and experienced by many users during talks and presentations without problems for music interaction. A log with all events sent through the application is available online through the main page of the web application in http://wscompmus.deusanyjunior.dj/cscores (visited on May, 2017). An example of an event is presented below:

> [4873] [2013-10-27 11:15:27 UTC] userid: 79 - score: s 0,168750 0,181234;d1;m13 0,166667 0,181234;m17 0,166667 0,181234;m72 0,172917 0,181234;m12 0,172917 0,181234;m17 0,172917 0,181234;m17 0,172917 0,181234;u16;

This event includes the timestamp of received event, the random identification selected for the user from the application, and the score. The score initiates with the xy initial position of the finger, includes the required delay applied before using the next finger position before each other finger position, and finishes with the delay before releasing the finger.

The development of Android applications with CSound was the topic of a talk during the regular seminars of Compmus at IME-USP, on October 1st, 2013[4]. The application was published as a technical paper in 2nd International CSound Conference (de Carvalho Junior, 2013) and is presented in Appendix F.2.

E.4 Android and Sensors: Sensors2

The creation of the *thereminimal* provided the necessary code for any person with minimum knowledge about Android programming to use the Android sensors. However, some users still found it hard to take advantage of all Android sensors in this case. The idea of creating an Android application that could make the use of Android sensors feasible started with Sensors2Pd and extended to Sensors2OSC, resulting in the conception of Sensors2[5] in a partnership with Thomas Mayer from Residuum.

The idea behind Sensors2 is to develop mobile applications that can help users to use Android sensors with many other technologies. *Sensors2Pd* aims the use of Android sensors with Pure Data patches that can be loaded on the application. *Sensors2OSC* sends Android sensors through the network using OSC format. *Sensors2Log* is an application that can save the sensors inputs to files inside device's internal or external memory. The applications are being distributed through the F-Droid[6] platform as well as through the github. A description of the development process of each application is present below.

[4] Talk about Android and CSound: http://compmus.ime.usp.br/en/node/377 (visited on May, 2017)
[5] Sensors2 web site: sensors2.org (visited on May, 2017)
[6] F-Droid website: https://f-droid.org/ (visited on May, 2017)

Sensors2Pd

Many mobile applications were developed in order to turn the mobile device into an instrument just as discussed in Section 2.3. Additionally, some applications provided the option to use Android sensors for music interaction, such as urMus, Control, and MobMuPlat. The main restriction of these applications is that they support only few sensors, while Android sensor options are increasing through the time. Based on this condition, *Sensors2Pd* was created following an approach that is able to support all current and future Android sensors.

The idea behind *Sensors2Pd* comes from the Android Sensors API[7]. Every type of sensor is registered at the API with an integer number. A list of these sensors and IDs is presented in Appendix B. The sensor manager used on Android applications provides a listener which informs the sensor ID and the latest values from an event received by it. Although it is possible to register to receive only the data regarding a specific sensor, the API offers the option to receive information regarding all sensors. In this case, *Sensors2Pd* provides an interface for sending the sensor's events to receivers at Pure Data patches. The receivers inside the patches must follow the patterns:

[receiver sensor{ID}v{#}]

The ID represents the sensor ID from the API documentation, and the # is the index of the event variable. Considering the accelerometer sensor that has the ID=1 and presents 3 variables representing the x, y, and z coordinates, the user needs to add the following receivers:

[receiver sensor1v0]

[receiver sensor1v1]

[receiver sensor1v2]

The application was programmed to send the values in a loop. This loop creates the messages automatically and send them based on the values that come from the API. This approach allowed the *Sensors2Pd* to be compatible with all available sensors at an Android device as long as the API continues with the same strategy.

Touch position is sent to Pure Data patches following a similar approach. The Android system offers a listener for touches that reports the x and y position of the touch on the screen including an ID for each finger. The IDs varies from 0 to 14, although Android devices support up to 10 touches. A finger receives an ID as soon as it is recognized on the screen, and these IDs start with 0 increasing as new fingertips are recognized. In case a finger is released and other fingers are still touching the screen, the next finger touching the screen will get the next available ID between 0 and 14. The receivers used for touches follow the pattern:

[receiver sensorTIDv#]

The ID represents the finger ID defined by the Android API, while the # represents the coordinate. An example of receivers for the finger with ID=0 are presented below:

[receiver sensorT0vx]

[receiver sensorT0vy]

Another sensor available at Sensors2Pd is the WiFi sensor. The Android API offers a lot of information regarding all available networks, such as SSID, BSSID, frequency, and signal level in dBm. In this case, the application sends SSID's level to Pd patches. The user needs to configure the receiver following the pattern:

[receiver sensorW-SSID]

[7] Android Sensors API header file from repository: https://android.googlesource.com/platform/hardware/libhardware/+/master/include/hardware/sensors.h (visited on May, 2017)

Table E.1: *Mapping from Android sensors to OSC messages in Sensors2OSC*

ID	OSC prefix	Dimensions
1	accelerometer	3
2	magneticfield	3
3	orientation	3
4	gyroscope	3
5	light	1
6	pressure	1
7	temperature	1
8	proximity	1
9	gravity	3
10	linearacceleration	3
11	rotationvector	4
12	relativehumidity	1
13	ambienttemperature	1
14	magneticfielduncalibrated	6
15	gamerotationvector	3
16	gyroscopeuncalibrated	6
17	significantmotion	1
18	stepdetector	1
19	stepcounter	1
20	georotationvector	4
21	heartrate	1
22	tiltdetector	1
23	wakegesture	1
24	glancegesture	1
25	pickupgesture	1

When the WiFi SSID is "MyRouter", the receiver needs to be:

[r sensorW-MyRouter]

The application was presented during the regular talks from Compmus at IME-USP on August 25th, 2014[8], and its code can be found in https://github.com/SensorApps/Sensors2Pd (visited on May, 2017). This code was also used during the development of the project Hoketus described in Section E.5. This application was published as a paper at the Joint Conference of the 40th International Computer Music Conference and 11th Sound and Music Computing Conference, Athens, Greece (de Carvalho Junior, 2014). This paper is presented in Appendix F.4.

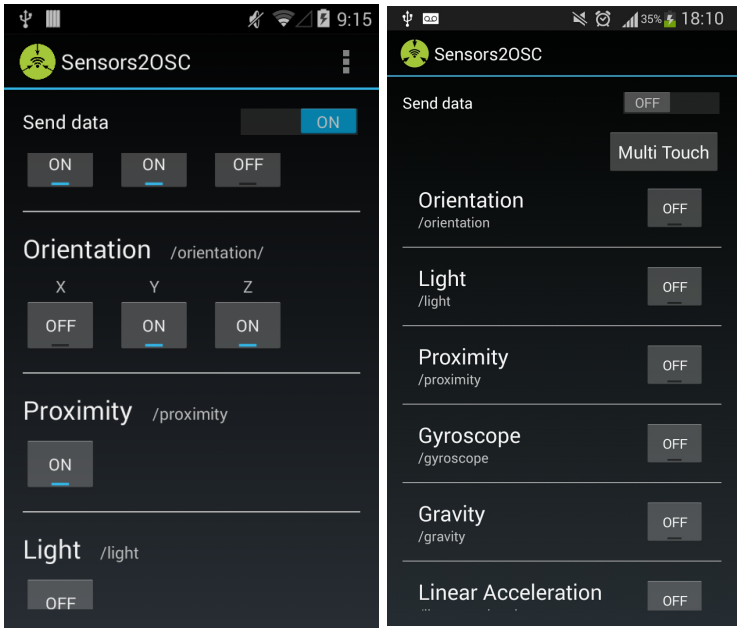
Sensors2OSC

The development of *Sensors2OSC* followed a strategy similar to the one applied during the development of *Sensors2Pd*. The application adapts to the available sensors at any Android device without restricting the use of new sensors that may become available. Sensor events are converted to OSC messages and sent through UDP to a IP and Port specified by the user on the settings screen of this application. The namespaces used for sending OSC messages have been updated for some sensors. Table E.1 presents the current personalized OSC prefixes. New sensors receive the prefix based on its ID. For instance, a new sensor with ID=30 will receive the prefix /30.

The first design of *Sensors2OSC* had the option to select the coordinates of each sensor that would be sent through OSC, as presented in Figure E.4a. In this case, one OSC message would be sent for every coordinate selected. As the UDP messages are received out of the expected sequence,

[8]Talk about Sensors2Pd: http://compmus.ime.usp.br/en/node/432 (visited on May, 2017)

some coordinates of a punctual sample can be lost even on local networks. This condition led to an update on the application in which the user selects only the sensor instead of its coordinates, and all coordinates are sent in a single bundle. The current version of *Sensors2OSC* packs all values and send them in a single OSC message, and its design is presented in Figure E.4b.



(a) *First screen design* **(b)** *Current design*

Figure E.4: *Screen-shots of Sensors2OSC main screen designs.*

The application has been used by Thomas Mayer in many of his performances, and I used this application in the performance *AntonioZ* at *Concertos de Computação Musical* (Computer Music Concerts)[9] organized by me in 2016 at the IME-USP during *Semana de Arte e Cultura da USP* (Art and Culture Week from USP). Other users have reported the use of this application and requested modifications. Mladen Radolovic is a Croatian user that requested the addition of an option to send NFC tags' information through OSC and now the application is able to send the NFC tags identification through the namespace /nfc. Tal Kirshboim is a German user that included the Multitouch option on the application and continue debugging the code. A user from Japan also contributed with the translation of the application to Japanese, and now the application is internationalized to four languages including English, German, and Portuguese.

A discussion about mapping Android sensors using OSC on Android devices with this application was presented during a talk in the regular Compmus seminars at IME-USP on June 29th, 2015[10]. The application code can be found in https://github.com/SensorApps/Sensors2OSC (visited on May, 2017). This application was published as paper in the Sound and Music Computing Conference at Maynooth, Ireland (de Carvalho Junior and Mayer, 2015). This paper is presented in Appendix F.6.

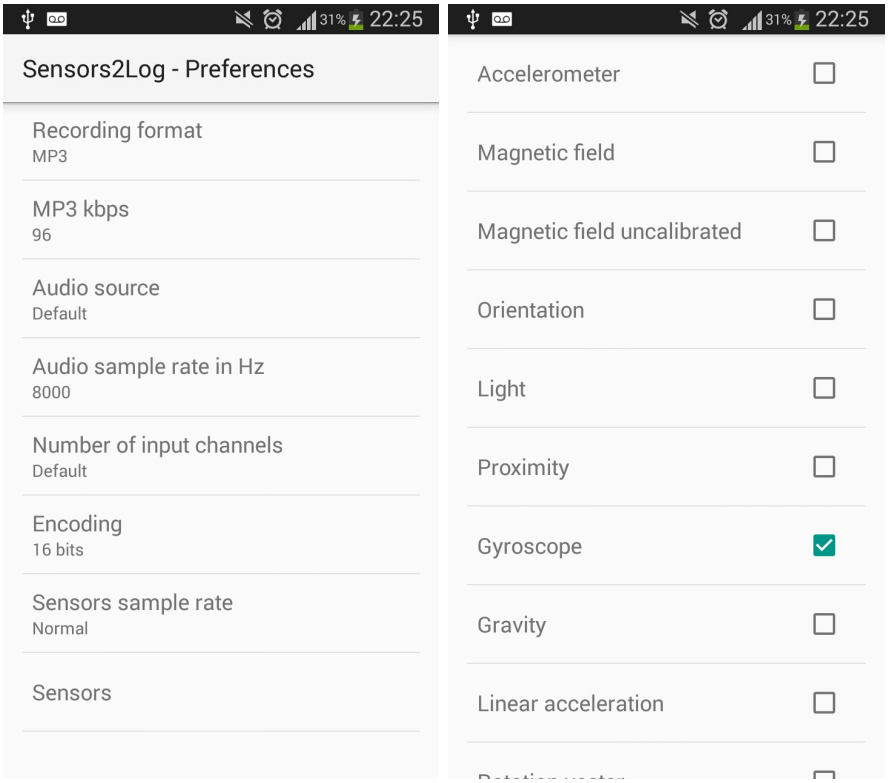
Sensors2Log

During the experiences with these applications I was invited to work in a project related to patient monitoring. The main goal of this project was to diagnose a person's health by analyzing the data registered while he/she is sleeping The code of Sensors2Pd was then changed to record the sensors values to files instead of sending to Pure Data patches. Another feature of this project was the audio recording into MP3 and WAVE formats.

[9] Concerts description: https://theantonioz.wordpress.com/ (visited on November, 2018)
[10] Talk about mapping Android sensors to OSC: http://compmus.ime.usp.br/en/node/465 (visited on May, 2017)

The current application is still under development, but it has been tested in many different situations. The first evaluation was for monitoring a patient in *Laboratório do Sono* (Sleep Lab) at *Instituto do Coração* (Heart Institute) from USP. A strap was designed to hold the mobile device over the body of a patient and allow the device to be in the same position of the patient. Audio and sensors were registered during a whole night of sleeping. The data was then retrieved from the mobile device through the internal memory using the USB cable and evaluated by many programs on Desktop computers. Other evaluations were conducted to register the sensor values and verify movements accuracy with success. The use of this application generated many interesting results regarding Android sensors. Although the application recorded data for more than 10 hours in a row, the battery consumption was less than 50 % in a Sony Xperia Play smartphone with the screen turned off during the evaluation.

After the experiments, the application was dubbed *Sensors2Log* and aggregated to the *Sensors2* collection. The application can be used in many monitoring process with options for selecting the sensors and the audio recording settings. Figure E.5 shows two screens of the application and the current application code can be found in https://github.com/SensorApps/Sensors2Log (visited on May, 2017).



(a) *Preferences screen* **(b)** *Sensor selection screen*

Figure E.5: *Screen-shots of Sensors2Log.*

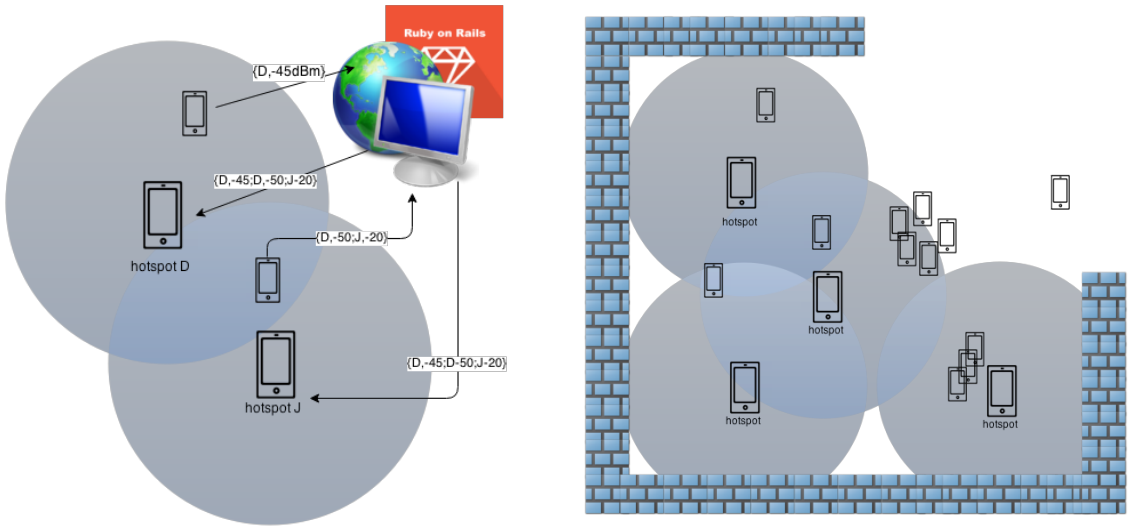
E.5 Indoor localization and Pure Data: Hoketus installation

The code of *Sensors2Pd* was used to create the application *Hoketus* during an installation under the same name. This installation took advantage of the WiFi level to identify users close to specific routers. The application was created with the Sensors2Pd code embedded and the application interface was personalized especially for this installation.

Some smartphones were configured as hotspots and the Pure Data patches created for this installation were defined to synthesize different sounds depending on the SSID found. The participants were invited to install an application that would find hotspots and also synthesize different sounds.

A web server was set up using the same technologies and approaches presented in Section E.3 but with specific pages for each hotspot. The participants' application had a method to send the level of the SSID found to the web service, and the hotspots' application were configured to request the SSIDs' level in order to amplify their synthesis depending on the number of close devices.

During the performance, the hotspots used the 3G interface to connect to the Internet, since the Wi-Fi interface was being used as a hotspot antenna. It is important to notice that the participants were possibly using the 3G data connection and nobody reported any issue regarding the consumption of the 3G plan during the performance. Most of the questions regarding the installation were related to interaction aspects that were somewhat incomprehensible at some locations. Figure E.6b represents the installation with the hotspots and the participants while Figure E.6a shows an example of network communication between the mobile devices. In both images the hotspots have a circle representing their best signal strength area.



(a) *Network communication* **(b)** *Four hotspots installation*

Figure E.6: *Hoketus installation.*

An aesthetic discussion about this installation and other experiences with Sensors2PD can be found at the paper presented at The Fourteenth Biennial Symposium on Arts and Technology, Connecticut College, New London, CT, USA (Bandeira and de Carvalho Junior, 2014). A discussion about the advantage of using WiFi for indoor localization with this application during the Hoketus project was published as a paper and presented as a poster during at the International Conference on New Interfaces for Musical Expression, Baton Rouge, Lousianna, USA (de Carvalho Junior, 2015). These papers are presented in Appendix F.3 and Appendix F.5, respectively. The code used on *Hoketus* mobile application is available in https://github.com/deusanyjunior/hoketus (visited on May, 2017).

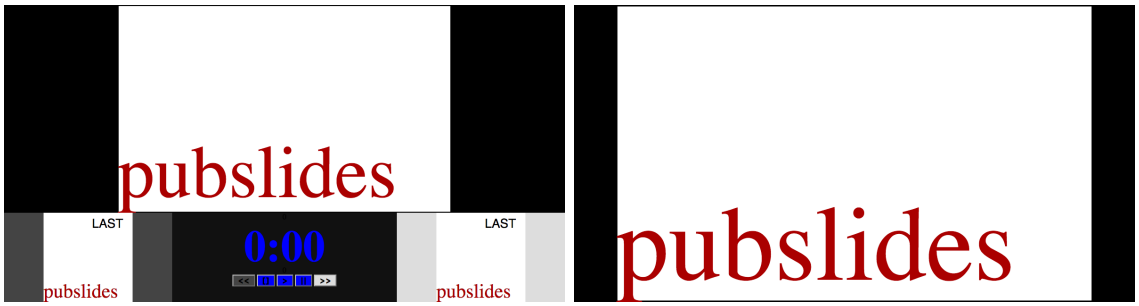
E.6 Cloud Services for presentations: pubslides

One of the first simple and useful application created with the use of Cloud Services during this research was the *pubslides*. This application was made with HTML, CSS, and Javascript with the aim of sharing the current slide page with the audience during a presentation. The person that is giving a talk can control the current slide position from the master page of *pubslides*, while the other users follow the presentation on the client page.

The process for adapting any presentation to this application is also simple. The *pubslides* code needs to be available at some web server accessible by the audience from the Internet, the presenter needs to open the master page presented on Figure E.7a, and the the audience needs to open the index page presented on Figure E.7b. The current version of pubslides require the PDF to be

converted to SVG files through a bash script. Every page of the PDF is then converted to an image, and this image is projected as the background image of a page at audience's screen.

The application is compatible with any device that has a browser, including smartphones. The communication between the master and clients takes advantage of PubNub cloud service. This cloud service was selected after some evaluations against Pusher cloud service. This comparison is better presented at the Chapter 5, but the main point considered was that PubNub offers a low packet lost rate even on free plans.



(a) *Master page* **(b)** *Client page*

Figure E.7: *pubslides application pages.*

This application is currently used at the *TV CompMus* page[11] used during the regular talks of the Compmus group. The code of the application is available in https://github.com/deusanyjunior/pubslides (visited on May, 2017).

E.7 Cloud Services and SuperCollider: SuperCopair

The conception of *SuperCopair* application happened during the period while I was an exchange student at University of Michigan. I was interacting the live coder Sang Won Lee and noticed that few solutions are available for a distributed collaborative live coding. After some research we discussed solutions and alternatives with Georg Essl and a the application presented in this section came out.

The *SuperCopair* application was created as a package to the Atom.io[12] IDE. Defined as 'a hackable text editor for the 21st Century' on its site, Atom is a text editor created using web technologies and has its development powered by the github community. This IDE has numerous packages for many programming languages and presents some solutions for coding, debugging, and managing projects. Atom packages are programmed in CoffeeScript[13], which is a programming language that can easily be converted to Javascript and can also integrate its libraries. The developers can install Atom packages to enable various functionalities in the IDE such as: communicate through chats, use auto-complete in certain programming language syntax, interact with desktop and web applications, integrate with the terminal command line, and have many options based on other packages. These features have motivated the development of SuperCopair package for Atom.

SuperCopair is based on two Atom packages: atom-supercollider and atom-pair. The first package turns Atom.io as an alternative SuperCollider IDE and permits users to openly communicate locally with SuperCollider audio server through OSC in the same way we can do on SC-IDE. Moreover, the users can take advantage of other Atom packages additionally to quarks packages. The latter package is used for pair programing through the Internet. The atom-pair package is based on Pusher cloud service and its default configuration is based on the community free plan, but a user can modify the settings and use the user's own keys within the personal free or paid plan. We decided to merge both packages to add new features for collaborative live coding, and finally had

[11] TV CompMus page: http://compmus.ime.usp.br/en/tv (visited on May, 2017)
[12] Atom.io website: http://atom.io (visited on May, 2017)
[13] CoffeeScript website: coffeescript.org (visited on May, 2017)

dubbed it the SuperCopair package. The main idea is that all participants have the opportunity to evolve into a collaborative performance.

The IDEs for SuperCollider have, by default, shortcuts to evaluate a line, a block, and to stop all sound process that is running. In addition to these options, the SuperCopair package includes methods and shortcuts to broadcast these events and execute them on all users connected at the same pairing session. Through the shortcuts, one can decide to evaluate selected code either only in the local machine, or in all computers of the session. One can also turn on a broadcast alert option in the settings in order to be asked before evaluating every broadcast event sent by another user in the same session. This allows each individual to have control over which portion of code can be evaluated in the local machine. The broadcast events are diffused through the cloud service and are expected to be evaluated as soon as each device receives the event message.

Networked live coding environments have been focused in collaboration. All participants are working together to execute a musical piece and share the same final result as a common goal. In most of the cases, sound is the only artifact that is shared, but the performers can also share codes. Some tools enable chatting, audio, and video interaction between the participants, whether they are sharing the same physical space or they are in different locations. Although the experiments with SuperCopair have been conducted as distributed collaborative live coding sessions, the cooperative environment has emerged during some moments.

During the sessions using SuperCopair, we had participants from different levels of experience on live coding and also some new users that decided to learn how to code during the session. The collaboration on the final sound was similar to other live coding sessions, but the experience added another way of interaction in live coding: the cooperation.

The cooperation aspect emerged from some events that came into view from the live coding sessions. Experienced users are faster and have written lots of code from the scratch without any problem, while the apprentices start from basic structures or from portions of codes available on the file. As all users were trying to synthesize the codes on all computers, it became easy to perceive if something was going wrong because everybody was sharing the code, evaluation, synthesis, and errors. The errors coming from novice programmers were often fixed by the experienced users, and they also discussed the solutions between themselves using comments on the file. Some tricks from the live coding practice were introduced by advanced users and the initial learners rapidly became instructors for new users and these instructors were following the same cooperative practices of the experienced users. These events are examples of how this application can be used in many different ways: for collaborative and also cooperative performances.

A discussion about the use of cloud services for designing collaborative applications was presented during a talk in the regular Compmus seminars at IME-USP on April 6th, 2015[14], and the discussion about the use of SuperCopair for live coding was presented August 19th, 2015[15]. The code of this application is available in https://atom.io/packages/supercopair (visited on May, 2017) with the documentation and environment setup instructions. The application was published as a paper in International Conference on Live Coding (ICLC), Leeds, West Yorkshire, UK (de Carvalho Junior *et al*, 2015a), which is presented in Appendix F.8. A discussion about using SuperCopair for cooperative live coding was published at Simposio Latinoamericano de Informática y Sociedad from the XLI Conferencia Latinoamericana en Informática (CLEI), Arequipa, Peru (de Carvalho Junior, 2015), and is presented in Appendix F.9.

E.8 Cloud Services and Web Audio: Crowd in C[loud] piece

An application for collaborative performance using smartphones was created with Sang Won Lee and Georg Essl while I was doing research at University of Michigan. The main idea behind this application is that it is fully deployed on the web, the communication between devices is conducted through cloud services, and the audio synthesis is made with Web Audio API. This combination

[14]Talk about cloud services: http://compmus.ime.usp.br/en/node/461 (visited on May, 2017)
[15]Talk about SuperCopair: http://compmus.ime.usp.br/en/node/469 (visited on May, 2017)

allowed participants to join the performance using any device and any kind of Internet connection as there was small messages transmission instead of audio.

As the name of the piece suggests, the crowd (audience) plays the musical instrument in C Major. This is directly inspired from the piece *In C*, by Terry Riley (Riley, 1964). In this piece, musicians (with various instruments) were guided to play pre-composed melodic fragments in sequence for a random amount of time. As it is up to each musician to decide how many times to play one fragment, the collective outcome of the ensemble creates a heterophonic texture of chance. Similarly, in *Crowd in C[loud]*, each audience member plays a series of short snippets composed by himself/herself and by other audience members. The interface provided will first guide a participant to compose a short "`tune`" that has five musical notes in C major.

Once the participant finishes the composition, he or she can browse, and play, what other audience members have composed. It is thus quite similar to Terry Riley's *In C*, in that one determines for oneself how long to play a tune. The difference is that there is no pre-composed fragments but each audience member will contribute to the piece by submitting a short melody. In this way, participants will have their own tunes and a chord scale become the common ground upon which the entire audience plays. In addition, there is a separate musician performing the piece on stage at the same time with the audience members in *Crowd in C[loud]*. The role of the musician is to be a meta-performer who can control the chord scale in which the audience members are playing. For example, the meta performer can, on the fly, change the instrument tuned in C major scale to a different chord scale (e.g., C Minor, Pentatonic Scale). This performer only controls the harmonic flow of the piece as generated by the crowd. The interplay between the musician and audience members ensures that each audience member will play individual patterns while a musician can progress the piece by changing chord scales. This performer-audience pairing model comes from a previous work of *echobo* (Lee and Freeman, 2013) where audience members played a simplified key instrument on smartphones with the chord progression determined by a performer on stage and synchronized over a mobile network.

The performance structure was inspired by other networked pieces that presented a centralized network with a server for receiving and sending information (Weinberg 2005). However, our choice of using the Pub Nub cloud service set us free from the burden of developing a custom server application, which is replaced with a single web page written in HTML and Javascript. This will be a useful setup for artists without networking background that want to write a network music piece with an interactive web page. PubNub cloud service permits easy interconnection between users through channels using publish-subscribe (or pub-sub paradigm). An application (or device) can subscribe to a channel and receive every notification that is published to the channel. For example to broadcast a message to all devices, a device can publish a message to a channel that every device is subscribed to. This provides a convenient abstraction that is robust against changes in network or end-user device and allows dynamic reconfiguration of participation. The push(or publish) notifications paradigm also makes performances more robust against technical disruptions such as disconnection and delays.

The left side of Figure E.8 presents the performance setup at the concert hall. There are two kinds of web pages running during the performance, one for the performer and another for the audience. The performer seat center-stage and his application (a web page) is projected onto a screen for the participants seated in the audience. The audience is instructed to visit the application that has the musical instrument on. The performance hall has wireless network available from the university and has a good reception of mobile network connectivity such as 3G.

A representation of the cloud service with the channels is on the right side of the Figure E.8. For this application there are three types of channels following the PubNub API for web application: a performer channel, an audience channel and individual channels equaling the participant number. The performer application is subscribed to the performer channel to receive messages sent from audience member's devices. Once a participant visits, it requests a performer's application to create an individual channel by publishing a message to the performer channel. The individual channel is then created to respond to an audience application's request, typically for the audience to retrieve a

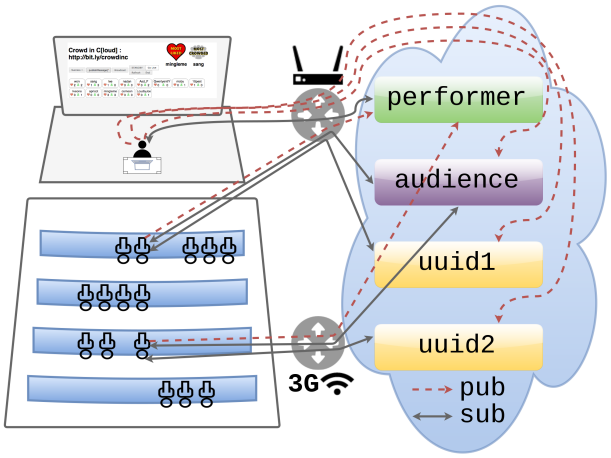


Figure E.8: *Performance structure. The left side depicts the performance space. On the right side, a rectangle represent a channel, solid lines show publishing and dotted lines show subscription.*

tune composed by others. When the page is loaded, every device is given a universal unique identifier (UUID) from the cloud service API, which is used to create an individual channel. Lastly, all the audience members are subscribed to the audience channel. This channel is used for crowd control: to send textual instructions, to live-code the mobile instrument, to start/end the performance and to troubleshoot the performance when it goes wrong (silence, refreshing the page).

Crowd in C[loud] was performed at many venues during the past two years and the debut happened on April 18, 2015 in Stamps Auditorium at the Walgreen Drama Center on the U-M North Campus[16]. The video from the debut is available in https://www.youtube.com/watch?v=8RWgXoM2BCA (visited on May, 2017) and a multi-camera video of a performance in Atlanta is available at https://www.youtube.com/watch?v=8nnrKJ4Ap0c (visited on May, 2017).

The discussion about sound synthesis was published as a paper in 2nd Web Audio Conference (WAC), Atlanta, Georgia, USA (Lee *et al*, 2016) and the discussion regarding network communication through the cloud services was published in the 16h International Conference of New Instruments for Musical Expression, Brisbane, Australia (de Carvalho Junior *et al*, 2016). The papers published in these conferences are presented in Appendix F.10 and Appendix F.11, respectively. A discussion about the application was presented during a talk in the regular Compmus seminars at IME-USP on September 16th, 2015[17] The code used on the application is available in https://github.com/panavrin/tindermusic (visited on May, 2017), with a live demo for the performer in https://crowdinc.github.io/performer (visited on November, 2018) and for the audience in https://crowdinc.github.io (visited on November, 2018).

E.9 iOS, Lua, and Network communication: Performances with urMus

The final projects of the course audited at University of Michigan resulted in musical performances. One of the performances that I participated during the conception was *Crowd in C[loud]* presented in Section E.8. Two other performances described below were based on mobile applications created using the *urMus* platform. Both performances used a similar approach regarding the network communication inside the platform using the Lua language.

Although the *urMus* supports OSC, listening to OSC messages through a specific IP is unavail-

[16] Debut of *Crowd in C[loud]*: http://www.eecs.umich.edu/eecs/about/articles/2015/w15-mobile-phone-ensemble-performance.html (visited on May, 2017)

[17] Talk about Crowd in C[loud]: http://compmus.ime.usp.br/en/node/475 (visited on May, 2017)

able so then the Multicast communication was discarded. The technology used in this case was the ZeroConf and Bonjour methods for network advertising and discovery. A Lua code simulating Multicast for urMus is presented and commented at Listing E.1. In a brief, the master device was configured to discover all other devices advertisements. Every discovered device is attached to a list of available devices. The master device can use this list to send messages to all devices.

```
friends = {} -- List with IPs from friends
friendsKey = {} -- List that associates the latest byte from friend's IP with
    its index at friends list

local myIP, myPort = HTTPServer()

-- The latestDot is the index before the latest byte from an IP.
latestDot = string.find(myIP, '.',
string.find(myIP,'.',
string.find(myIP, '.', 1 , true)+1, true)+1, true)

-- The id will be the latest byte from the IP
local ownId = tonumber(string.sub(myIP, latestDot+1))

-- Call this function when a device joins the performance
local function NewConnection(self, name)
DPrint("new friend "..name)
DPrint("")
for j,u in pairs(friends) do
if u == name then
return
end
end
friendLatestDot = string.find(name, '.', string.find(name, '.',string.find(
    name, '.',1, true)+1, true)+1, true)
friendId = tonumber(string.sub(name, friendLatestDot+1))
table.insert(friends, name) -- Adds IPs to the list friends
friendsKey[friendId] = table.getn(friends) -- Sets the IP index to the lates
    byte
end

-- Call this function when a device leaves the performance
local function LostConnection(self, name)
for l,w in pairs(friends) do
if w == name then
table.remove(friends,l)
end
end
end

-- Call this function to send a message to every participant device
function sendNote(noteToSend)
DPrint("sent "..noteToSend)
DPrint("")
-- Send the message to every device on the list
for indexIp = 1, table.getn(friends) do
local ip = friends[indexIp]
SendOSCMessage(ip,8888,"/urMus/numbers",noteToSend, ownId, currentPage)
end
end

-- Call this function when a packet is received from network
function gotOSC(self, noteReceived, senderId, senderPage)
DPrint("got osc")
DPrint("")
...
end
end
```

```
-- Call these functions when a friend joins or leaves the network
bg:Handle("OnNetConnect", NewConnection)
bg:Handle("OnNetDisconnect", LostConnection)

-- Listen and receive messages in the same Bonjour group
StartNetAdvertise("ttss",8889)
StartNetDiscovery("ttss")

-- Call this function for every OSC message received
bg:Handle("OnOSCMessage",gotOSC)

SetOSCPort(8888)
host, port = StartOSCListener()
```

Listing E.1: *Example of Lua code used at urMus platform for Multicasting in LAN*

This approach was evaluated for different numbers of devices and the more devices we added the more the application was overloaded with the network communication. The problem happens due to the loop throughout the list of IPs for every message occur inside the device instead of a loop at the router.

The pieces using this code are described in the next sections. The performances were performed on April 18, 2015 in Stamps Auditorium at the Walgreen Drama Center on the U-M North Campus[18]. The videos from these performances are available in https://www.youtube.com/playlist?list=PLcDYUKIkEOpZ9qMRzlfEg2ziEeJxBG4dF (visited on May, 2017).

Twinkle, Twinkle Smartphone Star

This performance was conceived in a partnership with Zachary Boulanger and the concept is related to playing with the idea of expectations. The master performer would be a composer using the master application with an 8-key keyboard that was configured to send every key touched to all the friends available. The other performers would have the same keyboard and they would receive the information regarding the notes to be played. Each note was converted to falling balls over the keyboard keys and the performers were invited to touch the correspondent key as soon as the ball reaches the other side of the screen.

On the performance conception, the composer expects the performers to play his creation exactly right, however the players soon realize they are unbound to this. The audience expects to hear well-known canons and to hear the music "written" by the composer, but once the performers start changing it up these expectations are broken. While composer is creating melodies based on canons, the audience will see the notes coming from the top of the projector screen and they are expected to hear the notes once they reach the bottom. In the beginning the performers will play only their specific notes, but some of them are going to try other notes and the perception of the melodies will diverge from the reality. As the performers decide to walk on the stage, it will became hard to know who is expected to play the right note.

The audience will expect some control from composer hands, but he is using a headphone and he is acting like a deaf composer. In the middle the composer will take out the headphone, he will try to control the performers again, and stop sending notes to the end of the piece. The sound experience comes from the idea that the audience will expect some melodies based on the notes that are falling down, and the players will play the notes at their own pace, without knowing the melody composed by the composer.

The code used for the performance is available in https://github.com/deusanyjunior/ttss (visited on November, 2018).

[18] Infomation about performances with urMus in 2015: http://www.eecs.umich.edu/eecs/about/articles/2015/w15-mobile-phone-ensemble-performance.html (visited on May, 2017)

Himalayan Singing Bowl

Biqiao "Didi" Zhang was the composer of this piece and I participated with Zachary Boulanger as partners during the development of the mobile application. In this piece we simulate a himalayan singing bowl through multiple mobile devices. One person is the "conductor"/"leader" of the performance and this person controls which note the devices are playing (e.g. C3) and how people should be moving their devices. Ideally, only the performers know who the leader is. The performers will all be standing in a circle on stage, facing inward. They move their devices up and down to control the volume and left and right to control the "beating" of the sound. The color on their screen changes with their movement. A projector displays a video of a singing bowl, so the audience knows what the performance is based on.

The difference in frequencies by each performer, their place on the stage, and the changing color of the screen mimicking their movements all provide a unique spatial element to the performance. The absence of a clearly defined conductor also lends to the idea of circular performance with everyone following everyone else's movements and actions. This shows that the performance not only simulates the singing bowls musically, but in a physical aspect as well. The sound is simulating an actual singing bowl, with the performers forming the bowl itself. Each person has a different frequency than those next to them, giving a spatial element to the musical experience. The performance is then made more musically interesting with the addition of different pitches, dynamics, and "beating". This is where our simulation grows and differs from the original, making the performance something new entirely.

The code used for the performance is available at
https://github.com/deusanyjunior/Himalayan-Singing-Bowl (visited on November, 2018).

E.10 Web Audio contribution to Open Band

Two researchers from NuSom group and Compmus created a performance named "Open Band" using many web technologies. The concept behind this performance was to create an online webchat and allow users to exchange messages publicly without nicknames or identification. The first version of this performance had a web page that converted letters into sounds and used audio samples. Although the performance was conceived to local networks, they were using a high number of audio samples, with 30MB of size. I attended some of their presentations and noticed that the performance was too heavy for mobile devices and the server could be overloaded depending on the number of participants considering that all participants were retrieving the audio samples from the server at the same time.

After some discussions, I decided to participate in a new version of the performance but changing some technologies. The current version of the "Open Band" is now entirely based on web audio synthesis and reduced the bandwidth requirements for the audience participation. This upgrade to the performance removed the restriction for performing in local networks and the system used can now be deployed on web servers.

During the development of this new version we opted to evaluate some ready-made solutions for web audio synthesis like Gibber (Roberts and Kuchera-Morin, 2012), Waax (Choi and Berger, 2013), and meSpeak.js[19]. These solutions are able to facilitate the use of web audio technology and other browser facilities, but also create barriers and sometimes limitations that are not imposed by pure web audio. Furthermore, sometimes frameworks could offer more than what was required by the application, which can lead to more confusion during development. We tried to use, meSpeak.js as a simple and interesting text-to-speech library but the resulting sound was not musically satisfactory, and the framework limited messages to play on top of each other. The current version focus on pure web audio for sound synthesis and we also decided to follow some aesthetic concepts based on phonetic and audio typography, to experiment with new kind of music, avoiding traditional chords

[19] meSpeak.js website: http://www.masswerk.at/mespeak/ (visited on May, 2017)

and scales. These concepts surrounded the whole development of this new version of "Open Band" and had a great impact in the new result obtained.

The code of the current version of the application used during this performance is available in https://github.com/fabiogoro/banda/tree/master/bandawa (visited on May, 2017). A discussion about the technologies used in this application is presented on a full paper published at the 3rd Web Audio Conference (WAC) held in August 21-23, 2017 at the Centre for Digital Music, Queen Mary University of London. The paper is available in Appendix F.12. A discussion about the performances using this application is presented on a full paper published at the Audio Mostly Conference held in August 23-26, 2017 at the Centre for Digital Music, Queen Mary University of London. The paper is available in Appendix F.13.

Appendix F

Conference papers published about this research project

F.1 SBCM 2013 - FFT benchmark on Android devices: Java versus JNI

Paper details

Title: *FFT benchmark on Android devices: Java versus JNI*
Authors: Antonio Deusany de Carvalho Junior, Max Rosan, André Bianchi, Marcelo Queiroz

Conference details

Title: 14o Simpósio Brasileiro de Computação Musical (SBCM)
Venue: Escola de Música de Brasília (EMB), Brasília, DF, Brazil
Dates: October 31 to November 2, 2013

FFT benchmark on Android devices: Java versus JNI

Antonio D. de Carvalho Jr[1] **, Max Rosan**[1] **, André Bianchi**[1] **, Marcelo Queiroz**[1]

[1]Computer Science Department – University of São Paulo

{dj,maxrosan,ajb,mqz}@ime.usp.br

Abstract. *This work presents a comparison of running times for Java and C/C++ implementations of the FFT algorithm on Android devices. We compare a pure Java implementation with the widely used FFTW library in C/C++, considering also the possibility of multi-threading. 35 different devices were benchmarked, and results on specific combinations of device model and operating system version are presented and discussed. We also discuss similarities between single- and multi-thread versions of FFTW on multicore devices and consider when developers can take advantage of each approach.*

1. Introduction

The Fast Fourier Transform (FFT) is an important algorithm for signal processing applications that can be used in many scenarios, for example, for creating tactile feedback by analyzing audio data (Lim et al., 2013), reducing noise on mobile voice communication (Jonathan and Leahy, 2010), performing face recognition (Cheng and Wang, 2011; Wang et al., 2010), and also for image processing on medical applications (Jonathan and Leahy, 2010). Since the FFT running time is $O(n \log n)$ (where n is the FFT block length in samples) (Cooley and Tukey, 1965), application development on devices with low performance may require fine-tuning some of the FFT internal details in order to ensure realtime operation.

FFTW (http://www.fftw.org/) is one of the fastest and most used FFT libraries (Lin et al., 2011), and its use on Android devices would appear to be a step forward for realtime signal processing compared to pure Java implementations on the Application level. Nowadays, multicore devices are becoming cheaper, thus turning multi-thread into a good approach both for developers and for users. On the other hand, since saving battery in mobile devices is also a primordial concern, using more than one processor just for the FFT algorithm is not an easy task. Another important consideration concerns devices' specific scheduling policies that might decide when to split processing into two or more cores or not. Multi-thread methods can give strange results depending on each device model internal peculiarities.

2. Methodology

This work presents the results of a benchmark of the realtime FFT on blocks of varying sizes using a Java implementation and the FFTW library, which is written in C/C++ and is called through the Java Native Interface (JNI). We have set up an environment to run arbitrary DSP algorithms over an audio stream segmented into blocks of N samples, allowing for the variation of algorithm parameters during execution. The software used is the Android DSP benchmarking application (Bianchi and Queiroz, 2012), an open source project available at https://github.com/andrejb/DspBenchmarking/, with some modifications to include the FFTW via JNI. To compare the performance of different implementations of the FFT algorithm, we considered a pure Java implementation, the single-thread

FFTW and the multi-thread FFTW. As the FFTW implementation is written in C, JNI was used to include the code into the benchmark. All performance measurements are made by the application started by the user. User interactive assistance is kept at a bare minimum, by starting the experiment and pressing a button to e-mail the results to the authors. To obtain as many results as possible, we launched an open call for participation through e-mail, and got responses comprising 35 different devices. Instructions were sent to stop all applications and turn off communication (tests could only be started after the user enabled flight mode) to impose an ”idle” scenario on every device. The result of imposing these constraints is an overall experiment that automatically cycles through all benchmarking algorithms, and then sends an e-mail report with results back to the authors.

3. Results and discussion

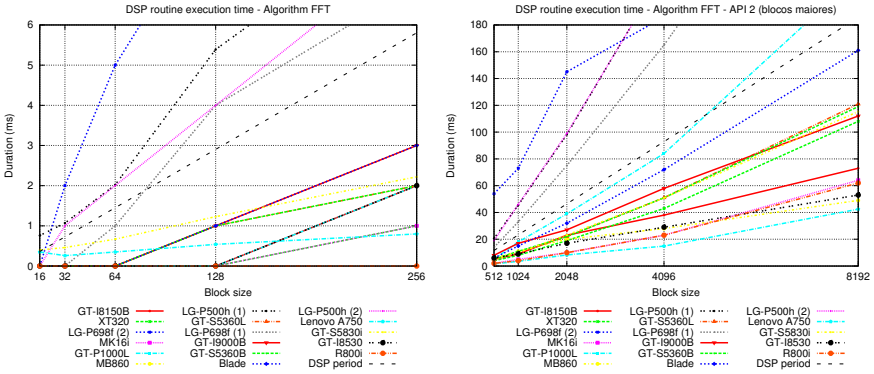


Figure 1: Benchmark of FFT implemented in pure JAVA in devices with API version 2.X for smaller (above) and larger (below) block sizes.

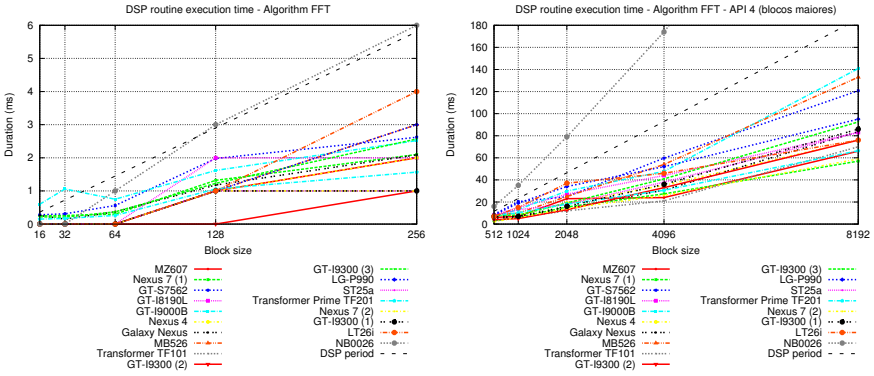


Figure 2: Benchmark of FFT implemented in pure JAVA in devices with API version 4.X for smaller (above) and larger (below) block sizes.

Figures 1 through 6 show the time taken by each of the FFT algorithms to be executed on each device as a function of block size. By comparing the Java FFT and single-thread FFTW results, on Figures 1, 2, 3, and 4 respectively, it is possible to notice that single-thread FFTW is faster than Java FFT on blocks larger than 16 samples. For N=16 the overhead of loading a dynamic library is not compensated by the efficiency of

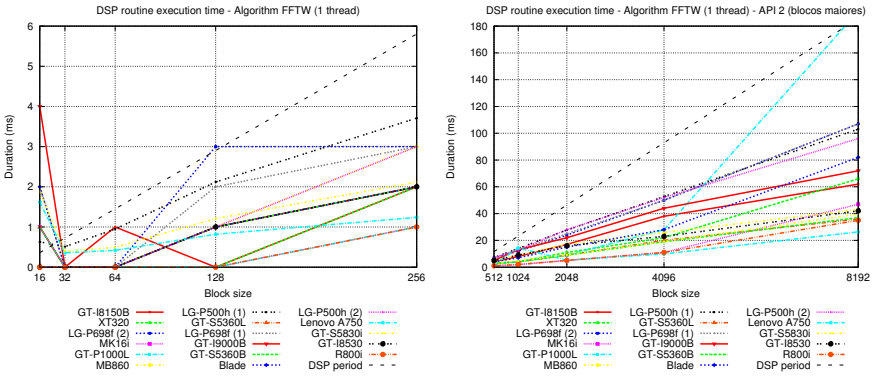


Figure 3: Benchmark of FFT using single-thread FFTW in native code in devices with API version 2.X smaller (above) and larger (below) block sizes.

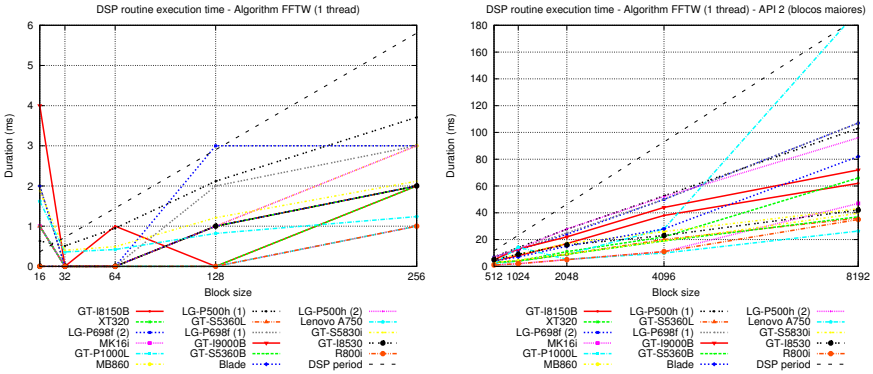


Figure 4: Benchmark of FFT using single-thread FFTW in native code in devices with API version 4.X smaller (above) and larger (below) block sizes.

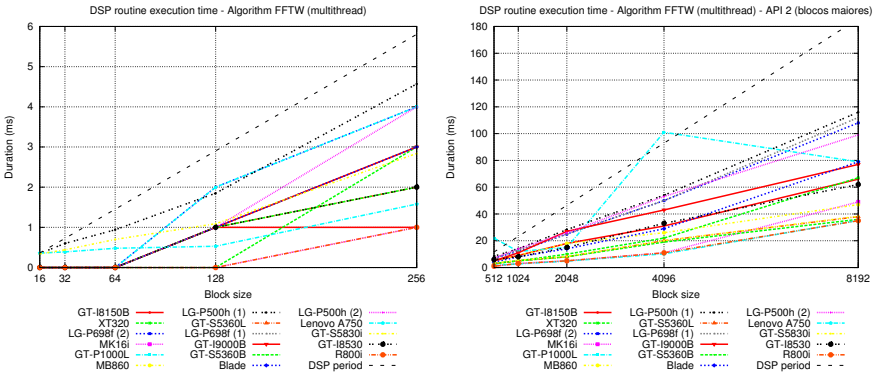


Figure 5: Benchmark of FFT using multi-thread FFTW in native code in devices with API version 2.X for smaller (above) larger (below) block sizes.

the C code, so that the results for the single-thread FFTW are actually worse than for the Java FFT, but for larger block sizes the library is already loaded. The same is observed with the multi-thread FFTW: it is executed after the single-thread FFTW and therefore

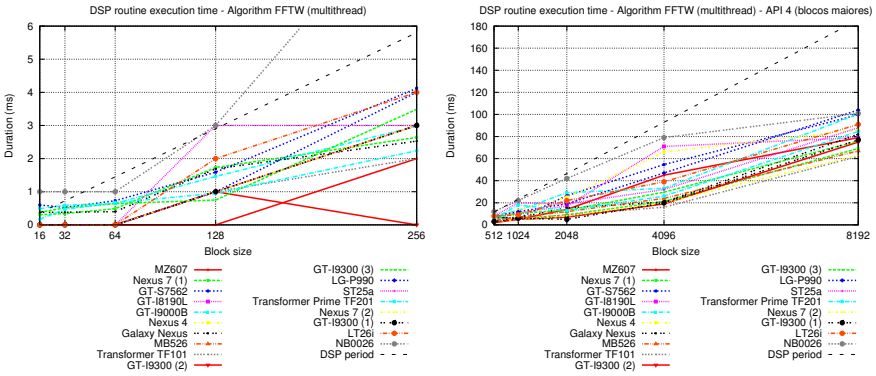


Figure 6: Benchmark of FFT using multi-thread FFTW in native code in devices with API version 4.X for smaller (above) larger (below) block sizes.

there is no overhead for the initialization of the library.

It is also possible to notice on Figures 5 and 6 that FFTW has worse performance when it is used with multiple threads. As it turned out, the Android kernel has different policies in different devices, and some devices move threads to different cores only when they are CPU-intensive and have been running for a sustained period. The use of native code does not automatically imply better performance on every situation. It had been noticed that using JNI increases application complexity and also has a cost associated with calls to non-Java code, which makes it indeed worse for some applications (Lin et al., 2011). Nevertheless, the implementation and comparison with native code for realtime signal processing evaluated in our work show that there are more subtleties that should be kept in mind, like the amount of data processed and the device specification about starting cores and moving threads around.

References

Bianchi, A. J. and Queiroz, M. (2012). On the performance of real-time dsp on android devices. *Proceedings of the 9th Sound and Music Computing Conference*, pages 113–120.

Cheng, K.-T. and Wang, Y.-C. (2011). Using mobile gpu for general-purpose computing: a case study of face recognition on smartphones. In *VLSI Design, Automation and Test (VLSI-DAT), 2011 International Symposium on*, pages 1–4.

Cooley, J. and Tukey, J. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301.

Jonathan, E. and Leahy, M. (2010). Investigating a smartphone imaging unit for photoplethysmography. *Physiological Measurement*, 31(11):N79.

Lim, J.-M., Lee, J.-U., Kyung, K.-U., and Ryou, J.-C. (2013). An audio-haptic feedbacks for enhancing user experience in mobile devices. In *Consumer Electronics (ICCE), 2013 IEEE International Conference on*, pages 49–50.

Lin, C.-M., Lin, J.-H., Dow, C.-R., and Wen, C.-M. (2011). Benchmark dalvik and native code for android system. In *Innovations in Bio-inspired Computing and Applications (IBICA), 2011 Second International Conference on*, pages 320 –323.

Wang, Y.-C., Pang, S., and Cheng, K.-T. (2010). A gpu-accelerated face annotation system for smartphones. In *Proceedings of the international conference on Multimedia*, MM '10, pages 1667–1668, New York, NY, USA. ACM.

F.2 ICSC 2013 - Touches on the line: Sharing Csound scores using web server and mobile phones

Paper details

Title: *Touches on the line: Sharing Csound scores using web server and mobile phones*
Authors: Antonio Deusany de Carvalho Junior

Conference details

Title: 2nd International CSound Conference (ICSC)
Venue: Berklee College of Music, Boston, MA, USA
Dates: October 25 to 27, 2013

Remarks

During the conference, I performed at the Concert #6 on October 27th with the application running on two mobile devices at the David Friend Recital Hall, Berklee College of Music, Boston, MA, USA[1]. This performance happened just before the concert in which performed Jean-Claude Risset, John Chowning, and Richard Boulanger, the masters of Computer Music that I met during this conference.

I also met two important woman for Computer Music during this conference. Sitting by her side during concerts, I had a brief talk with Marjorie Matthews, the spouse of Max Matthews, the father of Computer Music. The other woman was Judy Klein, a pianist and professor of electronic music. We went out for a cup of tea and she exposed me the great history of Computer Music during the 1980's when she was working with the first versions of CSound and waiting the whole night to listen minutes of synthesized sounds.

[1] http://www.csounds.com/csound-conference-poster-schedule/

Touches on the line: Sharing Csound scores using web server and mobile phones

Antonio Deusany de Carvalho Junior[1]

[1]Instituto de Matemática e Estatística
Universidade de São Paulo, São Paulo,Brasil

dj@ime.usp.br

Abstract. *This paper presents the results of an experiment which was based on an application developed for Android containing Csound embedded and sharing scores among multiple users during a distributed musical performance via the Internet using a web application in Ruby on Rails and hosted as web server. Results detail are described in order to present new perspectives for developing applications aiming to collaboration on mobile performances.*

1. Introdução

A variety of digital content is distributed through the Internet every second in order to provide interaction opportunities between people and contents in a highly collaborative reality [Mather et al. 2009]. The flow of content has increased so much that lots of content manager systems have been developed to make faster editing and delivery of content between users [Brampton 2008]. Multimedia content can also be available using these systems and increase the possibilities of the interaction practices[Chapman and Chapman 2000].

In this paper, we discuss technologies related to data communication over the Internet, focusing on multimedia data, or Csound scores, specifically. We also present results from test using an application developed to exchange scores between mobile phones connected to a web server . The test results will be discussed with the aim of promoting the use of such technologies for the development of other ideas involving content manager systems and musical interaction with multimedia technologies.

2. Technologies for content delivering

There are several ways to exchange data and make content available on the Internet. The most common technology used on this case is the web server, nowadays. The development of web server have been intensified over time so that new methodologies as Rapid Application Development (RAD) began to be practiced with the use of tools and specific frameworks. One of the technologies that emerged following the RAD methodology was Ruby on Rails (RoR)[1]. Rails is an open source framework for web applications that supports Ruby language.

RoR provides different ways for user's communication with the web server, among which some specific format of message can be exchanged, and here we have chosen JSON format as the preference. JSON is an open standard for exchanging data that is

[1]http://rubyonrails.org/

based on a text format understandable by humans without technical knowledge of computer science. The format is derived from the Javascript language and presents itself as an alternative to XML format. Sharing content with web server using this format comes to be advantageous also with the ease of encoding and decoding messages.

Web Server and content delivering have been heavily used by diverse kind of application, with special focus on mobile applications. Among mobile technologies, Android operational system has become widespread and several companies have adopted it into their devices. It means that a software developed for Android devices will have fewer restrictions for running whether you have a Sony or Samsung mobile device even from different versions of Android, despite distinct hardware specification used by both companies.

Those technologies are generically disposed to any kind of content, and its use in the music can open several possibilities for sharing content and predispose infinite ways of musical interaction [Iazzetta 2009, p. 123]. Facilitating its use by musicians may not be so simple, but we are going to present some examples in the next section that can illustrate possibilities of integration between new technologies with Csound and music expression.

3. Integrating technologies for sharing Csound scores

There are various possibilities that arise from individual use of each technology mentioned and also from their interactions. In this paper, an easy example is presented using all of them in conjunction with Csound. Both experimental music and non-realtime applications can benefit from user's interaction focusing on ubiquitous and pervasive computing concept. Bearing all of this in mind, a web application had been developed in a RoR web server, and also an Android application that uses that web application as principal way of communication.

Creating an RoR web application to respond to POST and GET web requests requires some commands described below:

```
$ rails new csoundwebapp
$ cd csoundwebapp
$ rails generate model Cscore user:string score:text
$ rails generate controller cscores index show new create
```

After these commands the application is created but some code needs to be edited on the RoR controller file in order the have the requests answered correclty.

File: */csoundwebapp/app/controllers/cscores_controller.rb*

```
class CscoresController < ApplicationController

  rescue_from ActiveRecord::RecordNotFound, :with => :record_not_found

  def record_not_found
      @cscore = Cscore.new();

      respond_to do |format|
        format.html # index.html.erb
        format.json { render :json => @cscore }
```

```
    end
  end

  def index
    @cscores = Cscore.order('id desc').all
    @cscore = Cscore.order('id desc').first

    respond_to do |format|
      format.html #index.html.erb
      format.json {render :json => @cscores }
    end
  end

  def show
    @cscore = Cscore.find(params[:id])

   respond_to do |format|
      format.html # index.html.erb
      format.json { render :json => @cscore }
    end
  end

  def new
    @cscore = Cscore.new

    respond_to do |format|
      format.html #new.html.erb
      format.json { render :json => @cscore }
    end
  end

  def create
    @cscore = Cscore.new(params[:cscore])

    respond_to do |format|
      if @cscore.save
        format.json { render :json => @cscore,
                      :status => :created, :location => @cscore }
      else
        format.html { render :action => "new" }
        format.json { render :json => @cscore.errors,
                      :status => :unprocessable_entity }
      end
    end
  end

end
```

The last commands needed are used to create the database where the scores will be saved and also to start the web app with the new configurations:

```
$rake db:migrate RAILS_ENV="production"$
$touch tmp/restart.txt$
```

The web application is completed and now the web server can be used. Consid-

ering the requests that can be sent to the web server, we have to code the POST in JSON format following the pattern:

{*"cscore"*=>{*"user"*=>*"username"*,*"score"*=>*"shared_csound_score"*}}

All request sent through POST are saved on databased for future GET request that will receive the same message as the score sent in JSON format. Supposing the web server address as '*http://www.server.web/*", the GET request for hundredth score needs to be sent to:

http://www.server.web/cscores/100.json

The Android application developed to exchange scores through the web server was based on Csound Android Examples developed by Victor Lazzarini and Steven Yi [2]. The MultiTouchXY application had been modified to play the user's score created by touch interaction and also send the same score to web server using JSON format. While that, the application send GET requests to web server aiming to receive new scores from other users playing at the same time. The last score sent to web server can be obtained from *cscore controller* using an specific request:

http://www.server.web/cscores.json

4. Tests and results

Tests were made using various specifications In order to evaluate the developed application. The web application was installed on a local server on a notebook, and also on a hosted server in the *DreamHost*[3]. The web server used was *Phusion Passenger*[4], which is free and was developed to easily integrate with Apache servers. Furthermore, this web server is recommended by RoR community as the best way to run web applications developed using RoR [**?**]. The devices used to test the application were *Sony Xperia Play* and *Tablet Multilaser Vibe NB026*.

The communication performance in both scenarios were quite different and we've got good results. On local server, the request latency had a variation from 5 to 500ms, while on hosted server the average latency for processing requests was form 100 to 1000ms. The main problem on local server was due to wireless half duplex connection specification, so the more devices sharing the same wifi connection the more latency we get. On the other hand, the hosted server permits each device to use different wifi connection to share the scores. Even with higher latency, the users can be connected from anywhere in the world and so the experimental performance can get valuable aesthetic characteristics.

5. Conclusions and remarks

As Android devices are becoming more available, the use of mobile applications during performances also become more practical, and the interaction with audience can easily be solved using such technology. The combination of several technologies can bring different perspectives to provide new applications previously unimaginable. Using mobile

[2] `http://sourceforge.net/projects/csound/files/csound5/Android/`
[3] http://www.dreamhost.com/
[4] https://www.phusionpassenger.com/

devices connected to the Internet to communicate through a web application on a web server can bring many solutions for problems that were difficult to be solved before.

The tests performed in this work put together some technologies in order to promote musical interaction between users in a ubiquitous manner in cases that there isn't dire need exchange of information in real time. The tests brought important results related to latency. Delays in communication have been found relatively low, while the worst problem came up to be on the local server environment, considering that wifi connection isn't full duplex as wired connection. So the best scenario for using this application model will depend on how many devices will share scores at the same time.

This work predispose a basic overview of what can be done to provide the use of free technologies for collaboration and interaction during performances using mobile devices. It is expected that several ideas and projects may arise from the solutions presented in this paper. Using all those technologies and putting them together in a harmonic way can benefit from user's interaction to monitoring systems, and considering that technological evolution tends not to stop, many other combinations are likely to be made ??in favor of such practices in the future.

References

Brampton, M. (2008). *PHP 5 CMS Framework Development*. Packt Publishing Ltd.

Chapman, N. and Chapman, J. (2000). *Digital Multimedia*. John Wiley & Sons.

Iazzetta, F. (2009). *Música e Mediação Tecnológica*. Perspectiva.

Mather, T., Kumaraswamy, S., and Latif, S. (2009). *Cloud Security and Privacy ? An Enterprise Perspective on Risks and Compliance*. O'Reilly.

F.3 BATS 2014 - Notes on the Elimination of the Mobile Music Audience

Paper details

Title: *Notes on the Elimination of the Mobile Music Audience*
Authors: André Damião Bandeira, Antonio Deusany de Carvalho Junior

Conference details

Title: 14th Biennial Arts and Technology Symposium
Venue: Connecticut College, Ammerman Center for Arts and Technology, New London, CT
Dates: February 27, 28 and March 1, 2014

Notes on the Elimination of the Mobile Music Audience

André Damião Bandeira
Escola de Comunicação e Artes
Universidade de São Paulo
andredamiao@usp.br

Antonio D. de Carvalho Junior
Instituto de Matemática e Estatística
Universidade de Sao Paulo
dj@ime.usp.br

Abstract

The aim of this paper is to present some strategies for creating works in Mobile Music that diminish, undermine or eliminate the barriers between artist and audience. Inspired by the text “Notes on the Elimination of the Audience” by Allan Kaprow. We will discuss this subject considering a public of music and sound art, illustrating it through some work examples and detailing the compositional process of two works of Mobile Music made by the authors.

Keywords: mobile music, participation, music performance, site specific, ubiquitous computing

Towards the end of Theatrical Conventions

In 1966, after a large dissemination of happening practices in New York, Allan Kaprow wrote the text “Notes on the Elimination of the Audience”, which lists some "rules-of-thumb" for this genre of performance. The main rule is: “It follows that audiences should be eliminated entirely. All the elements - people, space, the particular materials and character of the environment, time - can in this way be integrated.”. The artist makes this statement aspiring that certain theatrical conventions would disappear, because these performance rituals would be a primary element for establishing the division between performer and audience. Nine years before, Guy Debord wrote in “Towards a Situationist International” ideas which resemble the affirmations made by Kaprow, applied to a different context, perhaps more broader, but pointing to a similar direction , “ The construction of situations begins on the other side of the modern collapse of the idea of the theatre. It is easy to see to what extent the very principle of the theatre - non-intervention - is attached to the alienation of the old world.”.

The dialogue between performers and audience has been largely expanded during the last decades through many forms of collective authorships, getting to forms of creative processes that would be unimaginable in the beginning of the twentieth century. However the field of music performance, including contemporary classical music, has conserved the models of rituals of the concert room: clapping, hand shakes, silence, coughing and, rarely, booing. As Seth Kim-Cohen

points out “in music as an academic, artistic, and performance discipline, there is a perceived need to identify—often to eliminate—aspects of production, reception, or discussion that are not specifically manifest in material form. ”. This approach “condemns” elements, which do not deal directly with the very basic core of sound (pitch, volume, duration, and timbre) and traditional music structuring, to the label of “extramusical”, putting it into a subclass of musical building blocks. Emphasizing the aspect which Debord highlights in non-participative situations, that is alienation, leaving to the audience only the role of contemplating the great masterpieces of music. Even through the use of new technologies in performances of live-electronic music, most of the time the interaction tends to be restricted to the instrumentalist and computer, leaving the audience outside the conversation, and if we consider interactive installations of multimedia art in most cases the interaction happens between the public and an interface. It is odd to disregard the possibilities that could be generated by incorporating the participation of the audience in music pieces, considering aspects of performance and the perception of sound. Specially through the use of new technologies. Some examples can be found inside and outside of the concert room, wandering around the fields of music and sound art. We believe that one powerful environment for experimentation with interaction and sharing with the public is the area of Mobile Music, for its potentialities of data processing, network capabilities, wide accessibility and a growing community of artist and developers.

Mobile Music and Participation

The definiton of Mobile Music might be as wide as as the concept of *Musica Mobilis* by Shuhei Hosokawa :“I define *Musica Mobilis* as music whose source voluntarily or involuntarily moves from one point to another, coordinated by the corporal transportation of source owner(s).”(HOSOKAWA p.166). Although, in practical terms it could be considered a music which any part of its compositional process is mediated by mobile devices and takes advantage of its portability. What could include stage performances with mobile apps, standalone mobile applications, algorithmic music that makes use of GPS data, portable interfaces, etc. What generates innumerable possibilities of performance and installation configurations. Two well-known crucial points for making Mobile Music so attractive and suitable for participative art are its networking capabilities, providing easy communication and peer-to-peer actions, and the built in sensors in tablets and smartphones, which are able to generate a great amount of data to be used in various structures of control.

The use participation in performance practices is a very delicate issue, depending on how the interactivity is placed to the participants. As the sound artist Georg Klein highlights “The

performer's relationship with the audience had already been approached and examined radically during the 1960s. Experimental concert forms, theatrical abuse of audiences and actionist performances broke down conventional audience attitudes, but their shock effect was also exhausted. ". Kaprow himself draws attention to the fact that the artist should prepare the audience beforehand, so they would know which role each participant would have, and that would be a form of showing mutual respect. This statement should definitely be taken into account, but we should also consider that the use of new technologies contributes to new forms of production and reception. Which, for example, the use of computers might add a more playful aspect. Debord foresees the implementation of new technologies as way to ease the participation, which he suggests for the firsts situationists actions: "Besides the direct means that will be used toward precise ends, the construction of situations will require, in its affirmative phase, a new implementation of reproductive technologies. We could imagine, for example, live televisual projections of some aspects of one situation into another, bringing about modifications and interferences. ". Maybe the idea of being "protected behind the screen" of mobile devices could be a form of inviting people in to the performance.

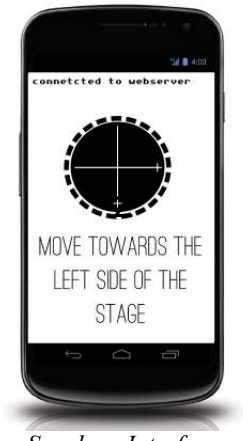
Inside the Concert Room

The concert room is certainly not the best place to execute a piece which requires participation. As we said previously the concert room is filled of rituals, what naturally affects our perception. For Mobile Music pieces it is not different, and the concert room physical disposition does not propose a dialogue. But some elements of it could be adopted, and corrupted. For this fact there are not so many Mobile Music participative works inside the standard contemporary paradigm of concert halls. The most prevalent choice in this configuration would be to use smartphones and tablets as a NIME (New Interface for Music Expression) to control sound synthesis, sampling and visuals. Using native sensors in the devices, such as accelerometers, compass and multitouch. There are a many artists working with this concept, like Atau Tanaka and Adam Parkinson, MoPhO (Mobile Phone Orchestra) and the DigiEnsemble. One seminal piece in the context of Mobile Music in concert rooms is Dialtones, by Golan Levin, Scott Gibbons, Gregory Shakar, Yasmin Sohrawardy, Joris Gruber, Jörg Lehner, Gunther Schmidl and Erich Semlak. This work, made in 2001, explores mobile phones as sound synthesizers and the disposition of the audience in space, generating a "crowdfunded acousmonium". Although the aspect of participation in Dialtones is quite limited, it does not necessarily take the public away from a mode of contemplation, but it brings the mobile phone into a prohibited place, where the participants focus on something that was merely functional, the ringtones. Another interesting performance that uses a simple artifice, using

mobile devices in the concert room is an excerpt of “Digital Hug” by the chinese artist Leung Kei-chuek, aka GayBird. The artist points a camera in the direction of the audience, which is projected to the screen, and asks for the participants to put the light of their mobile phones facing the camera. Then GayBird turns a step-sequencer on, which plays a sound according to the luminosity read by the camera, and so the public can move during the performance and change the sequences executed by the algorithm. It is a very simple maneuver, but that public reaction exemplifies quite well the idea of the comfort of “being hidden behind the screen”. On the next topic there is an example of a project which we developed for the context of Mobile Music in the concert hall.

Falantes (Speakers)

Speakers is a piece for electronic piano, projection, wifi router, mobile devices and participants, to be presented in a concert room. The piece begins with a pianist improvising on the piano, but no sound comes out. Behind the pianist there is a projection showing a network address, which the public should access through their mobile devices. After connecting to the server, the audience should download an application, which connects them automatically to the network and starts receiving the OSC data emitted by the piano.The app has some synthesizers that are randomly chosen in the moment that it is opened, enabling different users to produce distinctive sounds. When the notes emitted by the piano start being received by the users' devices the sound begins to be spread around the concert hall, and all gestures made by the pianist acquire another meaning. During the performance some messages are displayed on the mobile devices screens, asking for the collaborator to move around the concert room, changing the position of the sound sources. These messages may have an absolute direction, “Move towards the right corner of the stage”, or a relative direction, “Join the player who is playing on your right”. The timbre and messages on the screen are modified according to the sensors input embedded in mobile devices.



Speakers Interface

What means that the participant starts playing a “new instrument”. Thus, piano and mobile devices become a single musical instrument, a meta-instrument, which is spread across dozens of sound sources around the room. Constituting a situation in which both, performer and participants, have to learn how to play in collaboration during the unfolding of the piece. We believe that this would be one way possible to subvert the relationship between performer and "audience". Stoping the public of being a passive and alienated element, to become fundamental part of the work.

Outside the Concert Room

Differently from creating works to be played inside the concert room, the outside world makes obvious that space and place are a crucial part of the piece, and that our perception is clearly affected by what surrounds us. The art work is exposed to aesthetic, sociological and political aspects in a much more crude and almost inevitable way, and the participant placed in a more complex situation. There are a lot of works with sound which explore the space and the walking as a form of remixing sounds, without the mediation of mobile devices, such as the Sound Walks by Pauline Oliveros, which is a very fruitful field for inspiration and reflection on how to create with pure sounds and the environment. These are not completely conceptual works, but leave hundred percent of the action to the participant. The same a idea of walking as an action of remixing sounds can be found on Electrical Walks, by Christina Kübisch, and Radioscape, by Edwin Van der Heide, in which both works make use of sonification of electromagnetic waves. But if we consider the fact that these are technology mediated works, they create a completely different condition of reception, comparing to the non-mediated works, which is more playful and musical. On the other hand, some works deal with the notion of the site specific. Klein applies the german word *Ortsklang* (*Ort* = address and *Klang* = sound, meaning the place or address of sound) to his works, that the author translates as SiteSounds, which express a stronger concept of site specificity (KLEIN). Creating this stronger relation with place the artists is able to compose relations with the participants in a much more intimate sphere, working with the deconstruction of our routine and everyday soundscape. The concept of *Musica Mobilis* by Hosokawa might be applied to both situations, considering that they explore the mobility the user and devices in space.

Hoketus

Hoketus is an interactive sound piece that intents to create augmented sound fields through the mediation of smartphones. Providing an environment for collective composition and an expanded listening of electronic sounds and the urban environment. The name is inspired by the

musical composition technique, hoketus, used in medieval polyphony, which is the alternation of very short melodic phrases or individual notes, between many instruments. The work is composed only of portable devices: handmade speakers, smartphones and two applications developed for android and iOS systems. Therefore, it is an itinerant installation/happening, which is easily transported and installed. The area of the work is defined by the location of 5 speakers, that may occupy areas ranging from 20 m² to 100 m², depending on the urban zone to be occupied. Each speaker has a built in smartphone, running an application that generates a pointillist rhythmic electronic sound, in which each speaker completes the other melodically, like the medieval hoketus, creating an autonomous sonic environment. One of the mobile phones allows users to access the network, and download the application which communicates with the chorus of speakers. From the moment the user enters the network, all smartphones built in the speakers, which are hotspots, start tracking the user's position through the quality of wifi signal.

Users interacting with the firsts tests of Hoketus.

If the participant draws into one of the hotspots the application starts to produce a sound rhythmically similar to the one emitted by the speaker, adding a new layer of sound, which starts to influence the sound of the speaker as well. The sound produced by the users application may be controlled through the touchscreen, which manipulates harmonic layers and records audio snippets captured from the environment. As it was sad before a second sonic layer is combined to the landscape, which is heard without the use of smartphones, revealing a hidden sound layer, private to

the listener, just like in other softwares of augmented reality that usually affect only visual aspects. In hoketus we aim for an augmented listening. But the action of the participant is not limited to his own ears, because his actions influences the sound of the speakers that are close to him, making the rhythmic phrases change to rougher and continuous sounds, which may be modulated through different types of sound synthesis that vary according to the user's velocity. This allows communication between multiple users simultaneously, making the space among speakers a large instrument, suitable for collective improvisations.

Conclusions

The existence of an audience was something that drove happening away from one of the ideals of Kaprow, which was to bring art to life praxis together. As it was pointed out by Pieter Burger, it was one of the failures of the post-war avant-garde. Differently from Kaprow we have no interest in trying to get art closer to life praxis, but to appropriate, sample, hack and rethink some gestures transposed from our daily mediated life to a performance-like situation. By turning the portable devices into instruments in the context of a social collaborative situation, in which meta-instruments could be controlled through many hands. Therefore, the concept of stage and the traditional music audience are barriers between the ideas of Mobile Music and the possibilities generated through the use of mobile devices.

REFERENCES:

Debord, G (1957). “Towards a Situationist International”. In Claire Bishops (Ed.) Particpation, MIT Press.

Hosokawa,S (1984). “The Walkman Effect”.Popular Music, Vol. 4, Performers and Audiences (1984), pp. 165-180, Cambridge Press

Jo, Kazuhiro And Tanaka, A. (2009) "The Music Participates In." In Franziska Schroeder (Ed.) *Performing Technology: User Content And The New Digital Media.* Cambridge Scholars Publishing

Kaprow, A (1966). , Assemblages, Environments and Happenings. New York: Harry N. Abrams

Klein, G (2009). Site-Sounds: On strategies of sound art in public space. Organised Sound, 14, pp 101-108. Cambridge University Press.

Kim-Cohen, S (2009). In the Blink of an Ear Toward a Non-Cochlear Sonic Art. Continuum.

F.4 ICMC 2014 - Sensors2PD: Mobile sensors and WiFi information as input for Pure Data

Paper details

Title: *Sensors2PD: Mobile sensors and WiFi information as input for Pure Data*
Authors: Antonio Deusany de Carvalho Junior

Conference details

Title: 40th International Computer Music Conference and 11th Sound and Music Computing Conference
Venue: Athens, Greece
Dates: September 14 to 20, 2014

Proceedings ICMC|SMC|2014 14-20 September 2014, Athens, Greece

Sensors2PD: Mobile sensors and WiFi information as input for Pure Data

Antonio Deusany de Carvalho Junior
Computer Music Research Group
Universidade de So Paulo
dj@ime.usp.br

ABSTRACT

This article presents the results of some experiments using mobile sensors in patches of pure data. We are considering normal sensor, connected devices and Android also focusing on information as hotspots available as our best improvement in this way. During testing was performed some communication through the web server, implementing push notifications, and we apply this approach in smart phone applications to react on interactive sound installations. Ambient sound is generated using Pure Data patches, and is collectively changed in the installation environment. Sensors2PD is a generic application that can load any Pure Data patch and can be freely used for other experiments. The description of the application and installation ideas are discussed based on technical results of our tests. Our approach is analyzed as a good concept for use in mobile music performances and installations, with clear advantages for sound and music computing.

1. INTRODUCTION

Users interaction on mobile music pieces have been improved on diverse modalities. Nowadays, it is common the scenario where user receives some instructions and normally uses some interfaces, web pages, or apps conditioned to a piece. On this situation, the user notices what is going to happen if some click is being done and how the things work. Corroborating with this idea, we think that this area have great improvement during the last decade. Another point is that it is becoming hard to add different interactions from the perspective of the new devices to applications, and specifically to Pure Data Patches.

Mobile is anywhere and everywhere, so we can expect a lot of mobile devices at every concert from classical to experimental. Advances on technology has increased the number of features in devices such as sensors and transceivers, taking advantage of features of nanotechnology and processing. Another major point to keep in mind is the evolution of the Internet connection. WiFi internet access is something must have in any public place, even though in some cases the service is not so good for a crowd. Moreover, to avoid interruption of Internet access at these sites, acquiring a GPRS/3G/4G plan is a condition for being always on. The devices are always looking for the best way to communicate over the Internet, and nowadays WiFi connections are chosen with a higher priority, because of its speed and range, considering that users save a lot of WiFi networks with automatic connection option checked. It is important to note that mobile devices continues searching (best) available Internet connection , even when connected.

Thinking of presentations of mobile music, lots of sensors and specific local information on mobile devices can be used in applications. Using sensors is attractive considering its variety and sensitivity range. Besides sensor importance, we became interested in the use of WiFi information that is reported over time. We also used the continuous Internet connection to exchange text information between users through web server.

Some works have already evaluated the use of WiFi Indoor location for robots [1, 2] and had good results: 1 meter accuracy, approximately. This observation can be useful for presentations inside where GPS will not work, and the use of WiFi information can not be easily perceived by users, if the mobile device is always in search of mobile networks, as already pointed. The fact that the phone is always connected the network is another important information that had taken our attention especially because during some performances we shared some information among participants through Wireless and had improvements in interactions without user perception too. Surely, heavy consumption of 3G for large flow of information can be a problem for some people, then it should be better to use short messages and send them sporadically to avoid running out Internet User data occurs. This approach may be useful for sending information and receiving notification through a web server and create new forms of interaction between users during presentations of mobile music.

We are going present the way we developed Sensors2PD, how easy we can send sensors informations to Pure Data patches, and explain the other possibilities we can get if we use sensors of WiFi and web servers on performance-related applications. In the end there will be an explanation about experiences with the application.

2. RELATED WORK

This area has a bunch of works related to mobile music installations for user interaction using sensors and web servers with mobile devices. The main focus of some works are becoming similar, and they discuss that HTML5 and Javascript on web browsers are the best solution for this kind of inter-

Proceedings ICMC|SMC|2014 14-20 September 2014, Athens, Greece

action. In SWARMED [3], the authors developed a framework for audience interaction during an installation accessing a website on captive portal. The novelty of using a captive portal is interesting on the aim of focusing user's Internet access only to the performance application during the presentation. On the other hand, the author notice that each participant would like to hear its contribution to the performance, but all the sounds are reproduced only on the central performer, and it is hard to give this feedback if lots of people is participating.

On NEXUS [4] project, that is related to distributed performance, authors attempt that a cross-platform application can really be conceived using dynamic web page, allowing server side web application to do the hard work while the participants interact using an interface placed into a browser on its own device. Even if both works use Ruby on Rails, NEXUS leaves the application on line for distribution and scalability facilities aiming a large number of participants from different places.

On the other hand, we have urMus [5], an environment for mobile application development. This is a directly related work that has an event-handling for mobile sensor events, with lots of features that permits easy musical application development. It is possible to load scripts and also share code between users over the network in some applications developed with urMus[6]. The only limitation for musicians is that Lua isn't used as Csound and Pure Data.

Another related work uses Csound as main *patch* language [7]. The users loads Csound orchestras and scores on mobile devices and uses interface options and some sensor events for interaction. This approach was a motivation for this work, and we also notice some advantages of a similar project, PdDroidParty [1] as inspiring for a new way of musical interaction using a common language for musicians and some transparent information presented below.

3. TRANSPARENT INFORMATION

There are lots of transparent information that can be used on mobile applications, and we are going just to cite some of them briefly:

- Available WiFi networks can be found everywhere, and the users are living the Always-On Era;
- The cost for send and receive short informations from web servers is becoming unimportant bearing in mind the price of 3G plans on most part of the world;
- Push notifications are being used by the most of the applications installed on mobile devices, aiming to keep the user updated, but always consuming users Internet connection as much as possible;
- The use of hot-spot informations to facilitate localization has been a used first with robots, but also by general systems like Google Maps.

Thinking about those premises we can conclude that we can extrapolate the usual limits of performances on a theater, with lots of mobile users, for example, considering "transparency" in the user point of view. Of course the user need to know that we are using those information, but it does not need to be explicitly during the performance.

4. SENSORS2PD

The integration of PD with Android application became possible with libpd, a library that grant loading patches and exchange messages using senders and receivers on applications. One of the drawbacks of this integration is a slight need of Java programming experience intended for Android application development. Apart from that situation, some native applications have been developed to smooth the way for non (Java) programmers use its own patch on Android devices. The most famous application with this concept is PdDroidParty. This application searches internal storage for patches and permits interaction using some interface options. Although it is a great solution, it does not facilitate the use of sensors with patches.

Android devices possess lots of sensible sensors that have been used on many situations. We notice that the quantity of sensors and its range vary based on device and also Android API, resulting in some restrictions during mobile performances if you want to use specific sensors or have different devices. Our approach try to solve these problems and presents other sensor's information that can be used.

4.1 Using sensors on Pure Data patches

Sensors2PD is an application that makes possible the use of all Android sensors and even more useful data from device functionalities. The application uses libpd [2] to load the patch that users can seek and select from mobile storage. Receivers for any sensor value can be used on the patch following the specification found on the guide. Depending on the sensor id (ID) and variable number (#), you need to configure the receiver like that:

[r sensorIDv#]

If you want to use Accelerometer, which has ID=1 and 3 variables, you can receive the sensor values using:

[r sensor1v0]

[r sensor1v1]

[r sensor1v2]

This is a reasonable approach considering that the application will be listening for any sensor variation and might send its updated value to the patch as fast as possible. Android sensors can be listened at four different rates, varying from $\tilde{0}$ to 200ms. The fast you retrieve values the fast you drain the battery on mobile application, but during some tests we've found out that it is not a problem to use the fastest mode if the normal time of a performance or installation won't be more than one hour. Despite the fact that not all the sensors are supported on all devices, it is not a

[1] PdDroidParty: `http://droidparty.net`

[2] libpd: `http://libpd.cc`

great problem. Only available sensors are going to be listened, and only the receiver for an unavailable sensor will not receive any value.

We considered each functionality that detects physical variations as a sensor, so other kind of sensors are being used on this application and can also be used with PD patches.

4.1.1 Touch events and position

All Android devices support multitouch interaction nowadays with high sensibility as it is the principal and the most intuitive way to access mobile options. We can manipulate up to 14 simultaneous touch ids to get noticed about its events and position on x and y axes, even if most mobile devices detect only 5 or 10 fingers. Based on the same procedure defined before, it is possible to use the position from every touch your device screen can detect. The receiver on the patch will need touch id (ID) and position coordinate x or y (#).

[r sensorTIDv#]

If you want to track first touch position, which has ID=0, you need:

[r sensorT0vx]

[r sensorT0vy]

4.1.2 Audio input

Audio input using [adc~] on Pure Data can have better results only when Pure Data is used as an Service on the application, and then we have opted to run libpd as background Service on our application. It is necessary to bear in mind that background operation continues to run even when you left the application or change to another one, like answering a call. In spite of this fact we preferred not to do anything on this case to prevent some installations to stop working while the application is running.

We do not need to worry about audio input and output between device, application, and PD while using libpd. On the other hand we can't control the audio configurations due to device specifications, so there is no guarantee about sample rate, block size, or number of channels. Notwithstanding that this can be important for some patches, we notice that audio quality was awesome on our tests, except for the delay between input output even on loop-back.

4.1.3 WiFi networks

Conceived by the concept of transparent information, we've decided to use the sensor of hot spots as another option for the application interaction. The sensor of hot spots gives lot of information as SSID, frequency, and level in db from network found. On our application we've decided to send the level as information for the PD patches. You need to configure the receiver with the SSID name (ID), that can't have any spaces.

[r sensorW-ID]

If your WiFi SSID is "MyRouter", you'll need the receiver:

[r sensorW-MyRouter]

Regarding that the WiFi level varies normally between -100 dBm to -1dBm depending on how far you are from the hotspot, you can use this value on many applications for indoor localization based on works from other areas of literature like robotic [1, 2].

The WiFi network signal has lots of problems with interference that turns the signal always unstable, but it still can be used as positioning system. There are 11 different frequencies for hot spots and the antenna needs to search each channel before returning a list of networks. Each request of available networks took 500ms to 2s on our tests and we noticed that some hot spots have minor problems during these search because of the orthogonality of channels 1, 6 and 11 frequencies.

4.1.4 Other sensors input under development

There is also other sensors being tested and implemented to add more possibilities:

- Bluetooth using SSID
- GPS using Google API
- Color identification using camera and OpenCV

We also consider that this application can be easily ported to other mobile technologies like iOS and desktop application. The integration of this application with other systems is not hard, and we also use some communication through web servers to send some sensors values to other applications. Informations about the web server are going to be described below.

4.2 Web server integration

Considering the intention to send and receive information between devices, the easiest way to implement was using web server. The development of web server have been intensified over time so that new methodologies as Rapid Application Development (RAD) began to be practiced with the use of tools and specific frameworks. One of the technologies that emerged following the RAD methodology was Ruby on Rails (RoR) [3] . Rails is an open source framework for web applications that supports Ruby language.

We have chosen JSON format as the preference for communication with RoR web server. JSON is an open standard for exchanging data that is based on a text format understandable by humans without technical knowledge of computer science. The format is derived from the Javascript language and presents itself as an alternative to XML format. Sharing content with web server using this format comes to be advantageous also with the ease of encoding and decoding messages.

Those technologies are generically disposed to any kind of content, and its use in the music can open several possibilities for sharing content and predispose infinite ways of musical interaction [8, p. 123].

[3] Ruby on Rails: `http://rubyonrails.org/`

5. EXPERIMENT AND INSTALLATION

We proposed an experiment to test the Sensors2PD at an interactive sound piece. We have created Pure Data patches using lots of sensors as input, and the users were advised to walk around a place to find some hot spots and interact with them. The sound output amplitude on mobile devices was controlled by the WiFi signal of the hot spots.

Other sensors were used to change sound velocity, pitch, articulation, and other expressive values. During the experiment, the application have sent hot spot information to the web server, so the hot spots knew how many users were near and could play different songs to interact with the public, using push notifications and transparent informations. Figure 1 shows people during the experiment.

Figure 1. People appreciating the installation

5.0.1 Technical discussion about Internet access

When a device is configured as hot spot, it can't check for available networks, but it is possible to access the Internet using other technologies: 3G, Bluetooth, Ethernet. It is important to notice that mobile devices differ in signal level when set as a hot spot due to their different antennas. Before this installation, we needed to check out the distance between hot spots to avoid the user find more than one good signal at the same time. In this case, we programmed to start the interaction only when the user find a hot spot with level greater than -40dBm.

6. CONCLUSIONS

The development of Sensors2PD predispose other users to create applications using Android sensors with Pure Data patches. As an original solution, we can say that the community can help to improve considering its distribution as open source code, and maybe it can be integrated to other applications as PdDroidParty and so many others. The ideas of installation presented here and using this application can be characterized as a start point to other installations and we wish the community can take advantages of our contribution as much as possible.

The use of sensors, transparent information, and web server push notification approach was evaluated with good results. There were lots of questions from users during the installation, and most of doubts were related to some interaction aspects that were incomprehensible for them at first sigh, due to the transparent information used. This condition shows some drawbacks and advantages of our concepts, indeed.

7. ACKNOWLEDGMENTS

We would like to thank CAPES and FAPESP for funding. The support of NuSom Research Centre on Sonology [4] and Computer Music Research Group [5] is really appreciated. We thank Marcelo Queiroz, André Bandeira and Flávio Schiavoni for suggestions and ideas. We also thank Antônio Barreto for his time and willingness to help.

8. REFERENCES

[1] J. Biswas and M. Veloso, "Wifi localization and navigation for autonomous indoor mobile robots," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 4379–4384.

[2] H. Liu, Y. Gan, J. Yang, S. Sidhom, Y. Wang, Y. Chen, and F. Ye, "Push the limit of wifi based localization for smartphones," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, ser. Mobicom '12. New York, NY, USA: ACM, 2012, pp. 305–316. [Online]. Available: http://doi.acm.org/10.1145/2348543.2348581

[3] A. Hindle, "Swarmed: Captive portals, mobile devices, and audience participation in multi-user music performance," in *Proceedings of the The International Conference on New Interfaces for Musical Expression*, ser. NIME '13, 2013.

[4] J. Allison, Y. Oh, and B. Taylor, "Nexus: Collaborative performance for the masses, handling instrument interface distribution through the web," in *Proceedings of the The International Conference on New Interfaces for Musical Expression*, ser. NIME '13, 2013.

[5] J. W. Kim and G. Essl, "Concepts and practical considerations of platform-independent design of mobile music environments," in *Proceedings of the International Computer Music Conference*, 2011, pp. 726–729.

[6] S. W. Lee and G. Essl, "Communication, control, and state sharing in networkedcollaborative live coding," in *Proceedings of 14th International Conference on New Interfaces for Musical Expression (NIME)*, Goldsmiths, University of London, London, UK, June 2014.

[7] V. Lazzarini, S. Yi, J. Timoney, D. Keller, and M. Pimenta, "The mobile csound platform," in *Proceedings of the Internationa Computer Music Conference*, 2012.

[8] F. Iazzetta, *Mica e Mediaoo Tecnolgica*. Perspectiva, 2009.

[4] NuSom: `http://www2.eca.usp.br/nusom/`
[5] Computer Music Research Group: `http://compmus.ime.usp.br/`

F.5 NIME 2015 - Indoor localization during installations using Wi-Fi

Paper details

Title: *Indoor localization during installations using Wi-Fi*
Authors: Antonio Deusany de Carvalho Junior

Conference details

Title: 15th International Conference on New Interfaces for Musical Expression
Venue: Lousiana State University, Banton Rouge, LA, USA
Dates: May 31 to June 3, 2015

Proceedings of the International Conference on New Interfaces for Musical Expression, Baton Rouge, LA, USA, May 31-June 3, 2015

Indoor localization during installations using Wi-Fi

Antonio Deusany de Carvalho Junior
Instituto de Matemática e Estatística
Universidade de São Paulo
São Paulo, Brazil
dj@ime.usp.br

ABSTRACT

The position of a participant during an installation is a valuable data. One may want to start some sample when someone cross a line or stop the music automatically whenever there is nobody inside the main area. GPS is a good solution for localization, but it usually loses its capabilities inside buildings. This paper discuss the use of Wi-Fi signal strength during an installation as an alternative to GPS.

Author Keywords

Mobile, Wi-Fi, Indoor localization, Installation

ACM Classification

C.2.6 [Computer Systems Organization] Computer - Communication Networks — Internetworking, C.5.3 [Computer Systems Organization] Computer System Implementation — Microcomputers, J.5 [Computer Applications] Arts and Humanities — Performing arts (e.g., dance, music)

1. INTRODUCTION

Lots of installations have permitted interaction through web sites, touch screen, camera, sensors, and other actuators, and used physical interaction or wireless technologies otherwise. Some works have already evaluated the use of Wi-Fi on indoor localization with robots [2, 4] and had good results with 1 meter of precision. One can consider that Wi-Fi interface is more suitable for indoor installations than GPS, because Wi-Fi interface searches antennas without draining the battery while GPS has a high power consumption and is not a reliable source of position in this case due to some constraints on indoor environments.

Every time a device searches for available networks, the user receives a list with level, frequency, SSID, BSSID, and security capabilities of all networks that were found. The level is a value measured in dBm and represents the signal strength of the network depending on the power of the transmission antenna, but it also relies on the distance between wireless devices. The Wi-Fi level varies normally between -100 dBm to 0dBm when the device is getting near a transceiver antenna.

The Wi-Fi network level has some stability problems including interferences caused by other wireless networks on the same channel. Mobile devices can have different protocols for communication like 802.11a/b/g/n and also 802.11ac, the newest one at the time of this paper. If the mobile device uses a different protocol from a wireless router, these devices won't find each other and cannot establish a communication. The devices need to search wireless networks using many channels and protocols, and the routers need to transmit data through many protocols in order to be compatible with a selection of devices. These facts imply that some devices will not find all the networks available all the time.

2. EVALUATION OF WI-FI QUALITY

Our evaluation was made with Sensors2PD, a mobile application developed by the authors to load Pure Data (PD) patches on Android devices and send sensor information to patch receivers on real-time [3]. You just need to create PD patches with receivers for sensors and use the application to load the patches.

We used the Wi-Fi SSID sensor and created a patch to use the level value as input on a receiver. An example of a patch used on our tests is presented on the Figure 1. In this example, every time the device receives an update about the networks available, the level of the network *eduroam* is sent to the receiver [r sensorW-eduroam]. As the level is always negative, its value is inverted before being evaluated as a MIDI note and converted to a frequency. In order to evaluate indoor localization with Wi-Fi, we set up some mobile devices as hotspots with different SSID and created PD patches with the Wi-Fi level attached to some the volume sliders. The hotspots were positioned with a minimum distance of 10m from each other and the decided to walk around with one mobile device and record the interaction with our testbed installation.

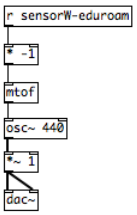


Figure 1: Example using Wi-Fi SSID sensor

We have noticed that each request of available networks takes from 200ms to 1s on these Android devices. In this case, the Java Virtual Machine is responsible for the frequency of this sensor and can control how many times this

Proceedings of the International Conference on New Interfaces for Musical Expression, Baton Rouge, LA, USA, May 31-June 3, 2015

information can be retrieved per second. It may be possible to reach better values using lower level calls. Another result was that we can only use the Wi-Fi signal as a real input when the level is greater than -45dbm. While the level is lower than this threshold, it became very poor and one may have frequent drops due to the oscillation of the values. An important point is that the position of the antennas will make a great difference while using routers.

3. INSTALLATION DESCRIPTION

We had a mobile music installation considering these results. A semi-open hall was the main area used to distribute the hotspots in a large area in order to permit lots of people to walk freely. We created an application using Sensors2PD with an embedded patch previously configured to interact with the SSID of our hotspots. The patch was prepared to play different sound patterns depending on the position of the participants on the main area and we also defined some regions with silent.

A Ruby on Rails webserver was created to receive and send data between the participants during the installation. We considered that most of the devices would be online during the installation and they could send some data through the Internet. The application was configured to connect to the webserver and send the list of Wi-Fi networks found by each device. On another point, the hotspots would pull down this data from the webserver and identify how many users would be walking nearby its own antenna. A PD patch was created to each hotspot with different music patterns. We used the 3G interface to connect the hotspots to the Internet, since the Wi-Fi interface was being used as a hotspot antenna. The Figure 2 represents the installation with the hotspots and the participants while Figure 3 shows an example of network communication between the mobile devices. In both images we have the hotspots with a circle representing its best signal strength area.

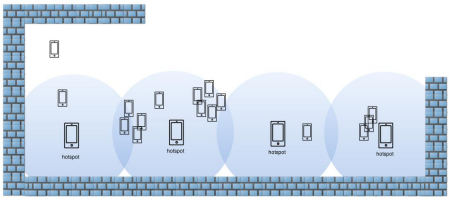


Figure 2: Installation with four hotspots

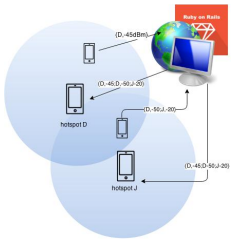


Figure 3: Representation of a network communication during this installation

At the beginning, the participants installed the application and started to walk around. We have noticed that during the installation most people was going to the same place as the crowd, and some groups were playing together near the same router. This social aspect was not predicted before the installation concept and may have affected the audience experience. Some participants were using earbuds and headphones during the installation, what made their sound experience personal in this way. An aesthetic discussion about this installation and other experiences with this Sensors2PD can be found at [1].

4. CONCLUSIONS

In this paper we have presented an idea to use Wi-Fi for indoor localization on mobile music installations. The main advantage of this approach is the fact that the audience can help with its own devices and there is no need to care about any additional artifact. Although GPS can be a solution for outdoor installations, we believe that Wi-Fi will fits better also in this way. Wi-Fi consumes less battery than the GPS system and we can assume that the Wi-Fi antenna will be always enabled on mobile devices due to the *always on* actual paradigm. Another advantage of Wi-Fi is that we may have 1m of precision [2, 4], instead of 10m that we have on GPS system. This precision is also important for performances on stages where someone can also implement the indoor localization using Wi-Fi.

The main drawback of our approach is the diversity of wireless antennas and protocols. We had some devices that could not interact during the performance, and we noticed that most of them had incompatible antennas with the installation hotspots. In this case, we can substitute the hotspots with routers that support all protocols in order to improve the number of devices that can interact with this kind of installation. Furthermore, the frequency of data information regarding the available Wi-Fi networks needs to be improved and we are still working on this point for the next version of Sensors2PD.

5. ACKNOWLEDGMENTS

The authors would like to thank the funding from Capes and FAPESP that made this research possible, and the support from the Research Centre on Sonology (NuSom)[1] and Computer Music Research Group (Compmus)[2].

6. REFERENCES

[1] A. D. Bandeira and A. D. de Carvalho Junior. Notes on the elimination of the mobile music audience. In *The Fourteenth Biennial Symposium on Arts and Technology*, 2014.

[2] J. Biswas and M. Veloso. Wifi localization and navigation for autonomous indoor mobile robots. In *International Conference on Robotics and Automation*, pages 4379–4384, 2010.

[3] A. D. de Carvalho Junior. Sensors2PD: Mobile sensors and WiFi information as input for Pure Data. In *Joint Conference: 40th International Computer Music Conference and 11th Sound and Music Computing Conference*, 2014.

[4] H. Liu, Y. Gan, J. Yang, S. Sidhom, Y. Wang, Y. Chen, and F. Ye. Push the limit of wifi based localization for smartphones. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, pages 305–316, 2012.

[1]NuSom: `http://www.eca.usp.br/nusom/`
[2]Compmus: `http://compmus.ime.usp.br/`

F.6 SMC 2015 - Sensors2OSC

Paper details

Title: *Sensors2OSC*
Authors: Antonio Deusany de Carvalho Junior, Thomas Mayer

Conference details

Title: 12th Sound and Music Computing Conference
Venue: Maynooth University, Ireland
Dates: July 26 to August 1, 2015

Sensors2OSC

Antonio Deusany de Carvalho Junior
Universidade de So Paulo
dj@ime.usp.br

Thomas Mayer
Residuum
thomas@residuum.org

ABSTRACT

In this paper we present an application that can send all events from any sensor available on an Android device using OSC and through Unicast or Multicast network communication. Sensors2OSC permits the user to activate and deactivate any sensor at runtime has forward compatibility with any new sensor that may become available without the need to upgrade the application for that. The sensors rate can be adjusted from the slowest to the fastest, and the user can configure any IP and port to set receivers for OSC messages. The application is described in detail with some discussion about Android device limitations and the advantages of this application in contrast with so many others that are available on the market.

1. INTRODUCTION

In the context of applications, we had already transformed mobile devices into many instruments, synthesizers, and controllers. Some applications present nice user interfaces with lots of knobs, sliders, and buttons, and are able to send MIDI or OSC to other devices with a new value set on the interface, acting as a general controller. These applications can suit users' needs in most cases, but the sensors available are almost the same all the time, and the users cannot decide if they want to get the values. This condition limits the usability and makes all devices acting as if they were the same.

The devices are upgrading in so many ways and music software is not able to follow their velocity. It is based on the famous race of hardware and software development, and as soon as new hardware is on the market, lot of applications are upgraded to support this new hardware in order to keep old users that are expected to buy this new hardware. Musicians and performers that are waiting for the new technologies are included in this group of users that will buy high tech devices as soon as they become available in order to experience its advantages and integration with old applications or devices, like digital audio workstations (DAW) and synthesizers.

A solution for this problem is to create flexible applications based on backward compatibility and forward compatibility concepts. The first concept is probably the most aimed and used on mobile application development due the support libraries available. However, the latter is not so simple to provide because developers will probably not know the next evolutions of hardware and cannot always test new hardware beforehand.

On the other hand, the Android API follow some development aspects that can permit some inference about the next updates on the codes. The SensorManager lets you access all sensors in the same way, and the data dispatched by each sensor are always represented in an array of floats. Furthermore, the sensors have an ID that does not change between devices or system versions. Sensors2OSC is an application that take advantage of these conditions and provide not only backward compatibility but also forward compatibility for the users of mobile devices. Additionally, Sensors2OSC uses OSC [1] to name each sensor with a human readable prefix and can be received by many languages and programs.

In the next sections we are going to present some related works that have similar functionalities. A precise description of Sensors2OSC is presented on Section 3, where one can find how to use and configure the application for any performance. Some case study are discussed in the end in order to invite readers to try this application even just to experiment.

2. RELATED WORKS

We have lots of applications using OSC on Android devices. Most of them can be used to create nice interfaces and send updated values from the widgets and controls using any OSC patterns. The lack of support for many sensors available on Android devices is a special limitation on all of them, and it has been requested by many users at online forums. We are going to present the most used and discuss the support of sensors available on all of them.

One of the most famous OSC application for Android devices is the TouchOSC [1] . This application provides a control surface for interaction based on many controls. The user can use some default interfaces that are included in the app, and we have the TouchOSC Editor for many operating systems that can be used to create any layout and customize the TouchOSC application depending on your use. Regarding the interaction using sensors, this application only supports accelerometer sensor and the values are sent continuously if enabled at the settings, so the user cannot control any option of this sensor from the main screen.

At the time of this paper, the highest rated application on both iOS app store and Google Play is Control [2]. Al-

[1] TouchOSC: http://hexler.net/software/touchosc

though similar to TouchOSC, Control uses web technologies and the user can create the interface using HTML, CSS, and Javascript. The interface design and configuration is described using JSON format, and it is possible to get the interface from the web or through OSC messages. The main limitation of this application is that it only supports accelerometer, gyroscope, and magnetometer (compass) sensors and cannot be easily upgraded due to its dependency of PhoneGap [2] libraries support for new sensors.

urMus [3] is an environment for mobile application development that also uses script language to create applications. This application has an event-handling for mobile sensor events, with lots of features that permits easy musical application development. It is possible to load scripts and also share code between users over the network in some applications developed with urMus [4]. This application supports the same sensors as Control, as well as GPS. The users can create any interface in Lua [3] language and send values through OSC. One can use any pattern of OSC prefix to send messages using urMus, the application can only understand its own defined prefix pattern.

MobMuPlat [4] is another example of mobile application that can be used to send the sensors values using OSC. Android devices can receive events accelerometer, gyroscope, orientation, compass, GPS, and joystick values. However, most of the options regarding the configuration of sensors are provided only for iOS devices.

Excluding TouchOSC, these applications are opensource and free. They represent the variety of available options for OSC users, but they are all limited to certain number of sensors. In the next section we will present a new solution for those who want to take advantage of all sensors available, independent of the device, including forward compatibility with every new sensor.

3. SENSORS2OSC

All mobile devices are coming with many new sensors, and the quantity and quality of the sensors are constantly increasing. On the other hand, it is not easy to use these sensors for musical interaction due to lack of good applications supporting all new sensors. We decided to create Sensors2OSC to solve these problems and permit the mobile device to be used in its full potential.

As the name may explain, the aim of this app is to get all sensors values and send through OSC. In this way, a performer can use any sensor from your mobile device to control applications on the same network. One can control many applications at the same time due to the support of OSC by many programming languages and programs, what implies in the requirement of adding only a receiver to a specific host and port in order to receive all messages sent by the mobile device.

Sensors2OSC has been created using Android Studio and gradle, the new pattern of mobile application development proposed by Google. The code is open source and has been tested in many versions of Android API, with help of many users. We support from Android API 8 (Froyo) to the newest ones, and we tried to design the app for forward compatibility regarding the sensors and the interface constraints. More details about the application are presented below.

[2] PhoneGap: `http://phonegap.com/`
[3] Lua: `http://www.lua.org/`
[4] MobMuPlat: `http://www.mobmuplat.com/`

3.1 Available sensors

Instead of limiting the application to only few sensors, we decided to load all available sensors at startup. This approach permits the use of all sensors available, and if a sensor will become available to the API in the future the user will have it enabled in the application. Some sensors have different number of values for each new event, and in this case we will have different OSC messages for each value.

Table 1 presents the OSC prefix for each sensor that can be available at Android devices at the time of this paper. The ID is the same ID used at Android API in order to identify the sensor. The prefix presented here are associated with the dimensions of the sensors. If a sensor has only one value, then the value will be sent as:

```
<osc_prefix> "f" <value>
```

If a sensor has multiple values, then the value will be sent as:

```
<osc_prefix>/<coordinate> "f" <value>
```

The coordinates are used by sensors with more than one dimension and their names differ depending on the number of dimensions as well. Sensors with 3 dimensions will have the coordinates `X`, `Y`, and `Z`; the ones with 4 dimensions are dispatched with `X`, `Y`, `Z`, and `cos`; and the sensors with 6 dimensions will have `X`, `Y`, `Z`, `dX`, `dY`, and `dZ`. These coordinate systems are modelled after the systems of currently available sensors.

The application is being updated for every new sensor added on Android API. We have also applied forward compatibility concept during the development. In this case, if you use this application on a device with a new sensor that is not already supported by our mapping, the ID of the sensor will be used as a prefix we are considering all new sensors with 6 dimensions. Another important point is that all sensor values are represented as float number from Android API, and that is the format used for all OSC messages sent through this application.

3.2 Main screen and controls

The main screen of the app has a main switch to start and stop sending values from enabled sensors using the configurations defined on the settings. All sensors available are presented on this screen and we have a switch for each coordinate of the sensor.

Once sending data for a coordinate is turned on, the app starts to send the OSC message defined for the specific sensor and coordinate, if the main switch is turned on. The user can turn off any coordinate at anytime. This functionality may be useful in many cases when you want to set a value and don't mind for next changes. As soon as you turn the switch on, the new events are going to be sent. It is important to notice that you won't receive the last state

ID	OSC prefix	Dimensions
1	accelerometer	3
2	magneticfield	3
3	orientation	3
4	gyroscope	3
5	light	1
6	pressure	1
7	temperature	1
8	proximity	1
9	gravity	3
10	linearacceleration	3
11	rotationvector	4
12	relativehumidity	1
13	ambienttemperature	1
14	magneticfielduncalibrated	6
15	gamerotationvector	3
16	gyroscopeuncalibrated	6
17	significantmotion	1
18	stepdetector	1
19	stepcounter	1
20	georotationvector	4
21	heartrate	1
22	tiltdetector	1
23	wakegesture	1
24	glancegesture	1
25	pickupgesture	1

Table 1. Mapping from Android sensors to OSC messages

if you turn on a sensor that had stopped to trigger events in the past, like the *tilt detector* that only trigger once and is disabled afterwards.

Figure 1 presents the main screen with some sensors. This screen is scrollable and you can see the osc_prefix to the right side of the sensor name. In case a sensor has more than one dimension, you will see the name of the coordinates that can be attached to osc_prefix. The switch in the top is the main switch and it has a fixed position, so the performer can attempt to stop or start the data transmission at any time, assuming full control at run time.

3.3 Network communication

The user can set the network details clicking at the settings button. It is possible to use Unicast or Multicast communication at this application. The mobile device with this application and the other device that will receive the values need to share the same network in case the user wants to send the OSC through Multicast. The range of Multicast address that can be used on IPv4 networks goes from 224.0.0.0 to 239.255.255.255 [5]. The Multicast has not been tested on IPv6 networks [6] with this application.

The Unicast communication requires a reachable IP address. It means that both devices need to be on the same network if the receiver is under NAT, or the receiver will need a reserved IP address on the Internet. Sensors2OSC supports both IPv4 and IPv6 address, and it is possible to use any hostname, like *localhost* or *example.com*.

The port number can be defined by the user at any time. However, it is recommended to use ports that are not reserved by any service. Some port number are assigned and some applications may have problems if the same port used for other purposes. The best port numbers to be used are the dynamic ports from 49152 to 65535 [7].

All of these settings described here are presented at Figure 2. It is also possible to define the sensor rate at this screen.

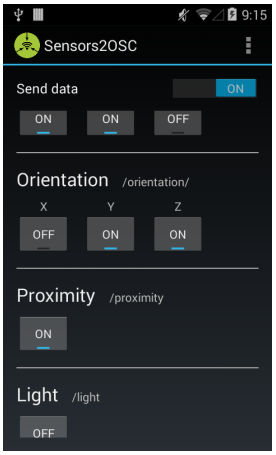


Figure 1. Sensors2OSC main screen.

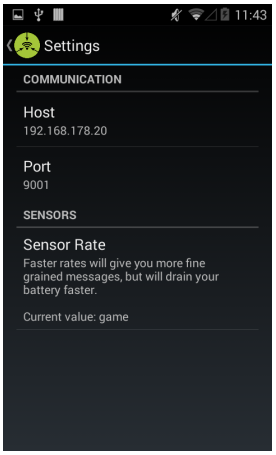


Figure 2. Sensors2OSC settings screen.

[5] Multicast addresses for IPv4 defined at IANA: http://www.iana.org/assignments/multicast-addresses/multicast-addresses.xhtml

[6] Multicast addresses for IPv6 defined at IANA: http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml

[7] Port number ranges defined by IANA: http://tools.ietf.org/html/rfc6335

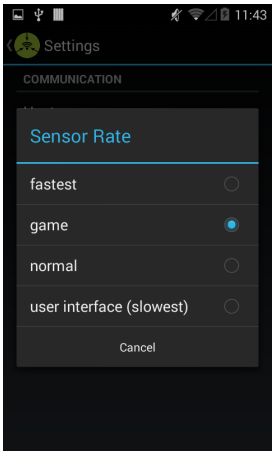


Figure 3. Sensors2OSC sensor rates screen.

Figure 4. Sensors2OSC help screen at Sony Z3 Compact device.

3.4 Sensor rate

The sensor rate defines the number of events that is expected by the user from each sensor. This rate is not fixed because it depends on the sensor changes and the system interruptions.

Some sensors trigger new events very fast, e.g. accelerometer, gyroscope, orientation, and magnetometer. Even when the device is disposed on a table, values for these sensors change due to the noise from the environment and sensor characteristics. In this case, the sensor rate can help to adjust the frequency of each new event that is going to be sent.

Other sensors have different operation mode. The proximity sensor will only change its value when an object is close to the device screen, and is expected to change again when there is nothing near the sensor range. Some new sensors are named motion sensors and they will have a one-shot event. In this case, the sensor will send a unique value before being disabled after that, and the sensor rate may not affect the events. In this group we have the sensors defined for gestures like the wake, glance, and pick up gesture, and also the significant motion and tilt detector.

The available rates are presented on Figure 3. According to Android API specification, the fastest rate will try to send a new event as soon as possible. The other rates are not well specified but follow the scale presented on the Figure 3.

Depending on device and requirements, the user may change the rate. All events will be packed and sent through the network without buffering, and CPU usage, and in turn power consumption may increase at this point. If latency is crucial, higher sensor rates will detect and send value changes faster. Balance between number of active sensors and the sensor rate need some attention, but the user is allowed to do whatever is wanted.

3.5 Help menu

The help menu is a recommended first option in this application because of its information. At this menu the user can find the number, the name, and the model of available sensors, and also the OSC addresses used for each sensor value, the sensor maximum positive value, and the resolution.

An example of a possible visualization of this screen is presented at Figure 4. The device used in this case is an updated version of Sony Z3 Compact with Android API 21 (Lollipop). This device has 18 available sensors from 25 supported by actual Android API. The range and resolution of the sensors will depend on the model of the sensor and may differ between devices.

4. CASE STUDY

The users can receive OSC messages in many applications, and some mobile devices are becoming cheap. In this way, it is possible to control one single application with two different mobile devices at the same time using Multicast.

Imagine a Theremin controlled by two mobile devices. One device can send the values from accelerometer to control the amplitude and the other device sends values from the orientation sensor to control the frequency. Both devices are configured to use the same Multicast address and the same port number.

In this case, the user will hold each device in one hand and control the sound moving the device around the three dimensional axes. Another idea is to do the same thing with two participants, so each participant can hold a device and control the amplitude or the frequency. The users can also change the controls deactivating one sensor and activating the other one. In case both users send values from the same sensor, the result cannot be predictable due to the indetermination of packet ordering on the network

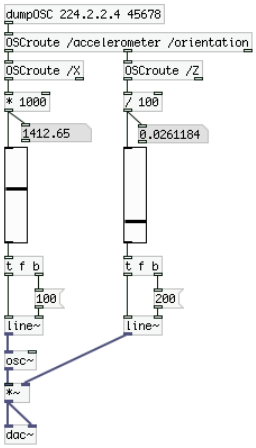


Figure 5. Case study of a Theremin controlled by two sensors and two devices.

while using Multicast and UDP packets.

A patch created in Pure Data to simulate this case study is presented on Figure 5. This patch receives OSC messages sent to Multicast address 224.2.2.4 and port 45678. The frequency of the oscillator is controlled by the X coordinate of the accelerometer sensor, while the amplitude is controlled by the Z coordinate of the orientation sensor.

An important information about the sensors are their ranges and resolutions. The user needs to verify these informations on Help menu before using the sensors at some application to adjust the values to its necessities. In this case study we had to adjust the values from both sensors to avoid bad values on the sound engine. The adjustment is optional and may vary also between the coordinates the sensors. For example, the orientation sensor have a range from 0 to 359 on X coordinate (the azimuth), -180 to 180 on Y coordinate (the pitch), and from -90 to 90 on Z coordinate (the roll). More information about the sensors can be found at Android developer web site [8].

5. CONCLUSIONS

In this paper we presented Sensors2OSC, an application that provide interaction using all sensors available on Android devices and OSC. This application is a sister of Sensors2PD [5], another application created by authors that sends sensors values to receivers on Pure Data patches that are loaded on mobile devices. The advances that we have with Sensors2OSC are the OSC format, Unicast and Multicast communication, sensor rates adjustment on the go, and a better nomenclature for prefix related to the sensors. Some of these options were suggested during the presentation of Sensors2PD in the last SMC conference.

All of these applications are member of Sensors2 [9], an attempt to create flexible applications that can send all events dispatched from sensors available on Android devices to applications using OSC, Pure Data, and other applications and languages like Csound, Processing, SuperCollider, and ChucK in a near future. At this point, we already evaluated the communication with Pure Data patches, SuperCollider, and Processing, however, we are planning performances using Csound, ChucK, and other systems.

A distant but important related work uses Csound as main *patching* language and accepts sensor values events at Android devices [6]. The users can load Csound orchestras and scores on mobile devices, uses interface options for controlling the sound, and uses accelerometer events for interaction. This application seems to be a proof of concept and is also an inspiration to the development of Sensors2CS using Csound as sound synthesizer in Android devices and receiving the values from all sensors.

Sensors2OSC is distributed with open source code that can be accessed on the repository [10]. The application has internationalization to English, German, and Portuguese. The authors are planning to publish the applications also on F-Droid [11], that is an installable catalog of Free and Open Source Software, and also on Google Play [12]. These resources may help users to learn how to use the sensors in their applications and can help non-technical users to install and use the applications without problems.

6. REFERENCES

[1] M. Wright, "Open sound control: An enabling technology for musical networking," *Org. Sound*, vol. 10, no. 3, pp. 193–200, Dec. 2005. [Online]. Available: http://dx.doi.org/10.1017/S1355771805000932

[2] C. Roberts, *Control: Software for end-user interface programming and interactive performance*, 2011.

[3] J. W. Kim and G. Essl, "Concepts and practical considerations of platform-independent design of mobile music environments," in *Proceedings of the International Computer Music Conference*, 2011, pp. 726–729.

[4] S. W. Lee and G. Essl, "Communication, control, and state sharing in networked collaborative live coding," in *Proceedings of 14th International Conference on New Interfaces for Musical Expression (NIME)*, Goldsmiths, University of London, London, UK, June 2014.

[5] A. D. de Carvalho Junior, "Sensors2PD: Mobile sensors and WiFi information as input for Pure Data," in *Joint Conference: 40th International Computer Music Conference and 11th Sound and Music Computing Conference*, 2014.

[6] V. Lazzarini, S. Yi, J. Timoney, D. Keller, and M. Pimenta, "The mobile csound platform," in *Proceedings of the International Computer Music Conference*, 2012.

[8] Android developer: `http://developer.android.com/`
[9] Sensors2: `http://sensors2.org/`
[10] Online repository of Sensors2OSC: `https://github.com/SensorApps/Sensors2OSC`
[11] F-Droid: `https://f-droid.org/`
[12] Google Play: `https://play.google.com/`

F.7 ICMC 2015 - Computer Music through the Cloud: Evaluating a Cloud Service for Collaborative Computer Music Applications

Paper details

Title: *Computer Music through the Cloud: Evaluating a Cloud Service for Collaborative Computer Music Applications*

Authors: Antonio Deusany de Carvalho Junior, Marcelo Queiroz, Georg Essl

Conference details

Title: 41st International Computer Music Conference (ICMC)

Venue: University of North Texas, Denton, TX, USA

Dates: September 25 to October 1, 2015

ICMC 2015 – Sept. 25 - Oct. 1, 2015 – CEMI, University of North Texas

Computer Music through the Cloud: Evaluating a Cloud Service for Collaborative Computer Music Applications

Antonio Deusany de Carvalho Junior, Marcelo Queiroz
Instituto de Matemática e Estatística
Universidade de São Paulo
dj,mqz@ime.usp.br

Georg Essl
Electrical Engineering and Computer Science
University of Michigan
gessl@umich.edu

ABSTRACT

Cloud computing offers a new level of very large scale distributed networking offering unprecedented level of networked participation and performance. This makes cloud services an attractive candidate for computer music concerts and applications. However, important network measures that are relevant for computer music performances, such as latency, have as yet not been explored in detail in this context. This paper presents results of an evaluation regarding the round-trip-time (RTT) for exchange messages through Pusher cloud service using mobile devices at different places on the American continent, exploring their use for very long distance collaborative performances. Average RTT varied from 230 to 578 milliseconds. Minimum round-trip time clocked in at 166ms in one of our tests between USA and Brazil. The details of our study can help develop collaborative network performances that can build strategies to incorporate expected latency patterns into either the back-end architecture or into aspects of the performance itself.

1. INTRODUCTION

New technologies can break the rules of old paradigms with possible solutions for classic problems. Cloud computing is an example of a tremendously successful solution for large scale problems that were not feasible before it. Cloud computing has gained confidence and popularity, and its use has become an easily accessible commodity. Although it is applicable for many computational problems, evaluation of the characteristics of cloud services for computer music applications is still lacking.

At first sight, the structure of the cloud is useful for computing scientific algorithms, but we can also make use of the fast data distribution between different locations. This approach can help many applications to behave as a responsive computer music environment for collaboration through the Internet. Another important point is that cloud computing creates an abstraction of the complex architecture that would be needed for an efficient communication between a large number of devices at the same time. Leveraging this abstraction makes the creation of very large scale networked performances substantially easier than before.

Musical collaboration often uses local and remote networks. Local networks have the advantage of guaranteeing low latency, making the technology relatively straight-forward to use in a performance setting, yet numerous long-distance Internet music performances have also been successfully conducted in the past. In most of those cases we notice that a specific network setup is necessary beforehand and the performers also have to manually configure network settings in order to suit their needs depending on many variables, such as the number of participants, the number of data streams and the exact nature of exchanged data. As opposed to that, balancing, scalability, and setup will be done behind the scenes on cloud computing solutions.

An important challenge of long distance networking comes from network latency in the communication between devices from different locations. A latency of 100ms and above implies some difficulty for network interaction even with experienced musicians [1], and if we have a sensitive ensemble performance, this threshold may be as low as 20ms [2]. One way to tackle this problem in local networks is using lightweight networking protocols such as UDP [3, 4], that has a low overhead and fast data diffusion. The potential for packet loss can often be of minimal impact to the ability to perform in practice.

In the case of long-distance networks there are no ready-made solutions, and performers may well have to grapple with the specific latency present in any given configuration. In order to facilitate this baseline understanding our work aims at evaluating the cloud computing latency and usability in the long-distance case.

We developed a mobile application using a cloud service for communication between mobile devices in different locations. We opted to exchange textual messages based on symbolic data that can be synthesized on local devices, avoiding audio transmission. One of our premises is that we can take advantage of mobile devices processing capabilities, that were already evaluated in previous works [5, 6]. This approach is also based on the heavy use of MIDI and OSC on

computer music performances, and its support by most programming languages used on mobile music applications.

Although most cloud computing applications require a configuration of virtual machines or web servers, we are exploring push-based cloud services as an alternative. These are widely used to service large scale distributed push-notifications in mobile apps. In particular we are using the cloud service offered by Pusher [1] for our work. Pusher makes the design of large scale data distribution easy through accessible libraries, providing a convenient gateway for developing scalable networked data distribution solutions.

In the next sections of the paper we will introduce related work before a discussion on cloud services and the Pusher cloud service. The evaluation and results are going to be presented in Sections 5 and 6, respectively, and we will conclude with the discussion and analysis of these results.

2. BACKGROUND AND RELATED WORKS

Experiments with network music date back to the 1970's with *The League of Automatic Music Composers* [7]. The set up of this group of composers was based on desktop computers and a local network. Each member would be responsible for some musical feature during the performance, and the interaction would take place through the local network, and later, through phone lines. An important pioneering example of mobile wireless networked performance is Golan Levin's "Dialtones" [8]. This performance, based on audience participation with mobile devices, was achieved by the definition of a ringtone for each participant on a determined seat, and the use of an application to call specific devices at any time. These kinds of music performances were only possible due to the creative use of new technologies available at each period, the commercialization of the personal computers and wired networking in the first instance, and popularization of mobile devices and wireless networking in the second case.

We have had an increasing number of important works during the 21st century, and this fact may be associated to the accelerated technological evolution of this period. Every new product or feature presented to the market is used on musical performances and installations soon after receiving musician's attention. Some examples are touchscreen sensors [9], cameras [10], mobile sensors [11, 5, 12], smartphones with all their features [13], laptops orchestras [14], and browsers with HTML5 and new Javascript libraries [15, 16]).

New communication technologies also encourage musical performances. Even advances in network protocol technologies and transmission media can alter and improve the possibility space for music performances. The protocols used on network music had great progress thanks to advances made in network protocols starting with TCP, and then UDP, SCTP, RTP, and RTSP [17]. Additionally, advances in communication regarding the quality of data transmission are impressive. Nowadays, one of the most popular transmission medium is the optical fiber used on gigabit Internet interconnection that reaches hundreds of billions of bits per second in some parts of the network.

The use of local network novelties can be seen on "Worldscape laptop orchestra" [3]. Here the authors decided to use the recent wireless standard 802.11n, due to its high frequency band and datarate that improved laptop communication if compared to old standards. Freeman's "massMobile" [18], Allison's "Nexus" [19], and Hindle's "SWARMED" [20] present different ways of using web servers and web technology for musical performances. In these works, web services fill the gap between user interaction and local installations or performances. An interesting aspect of "massMobile" [18] is the use of 3G and 4G technologies to ease audience participation.

Cloud computing suggests itself as the next logical step in network capability, ready to be used for music applications and performances. The "CloudOrch" [21] as proposed by Hindle is one of the first attempts to use the advantages of cloud computing in musical ways. The idea was to deploy virtual machines for client and server users, create websockets for intercommunication, and stream audio from cloud instruments to both desktop computers and mobile devices using web browsers. The author dubbed it "a sound card in the cloud" and presented latency results from 100 to 200ms between the Cybera cloud and the University of Alberta, Canada, using the HTTP protocol. Although similar, our work took full advantage of the cloud computing resources and focused on cloud services that allow the use of virtual machines with a high level of abstraction while retaining comparable performance.

3. CLOUD SERVICES

An important core aspect of cloud computing relies on an abstraction of distributed servers in order to simulate a centralized network with resource replication and balancing. Early on cloud computing was mostly used for scientific calculations, grid computing, or large scale computing. All these application areas required substantial computational power and hence benefited from many cores.

The cloud computing evolution pointed out the possibility of cloud services as a solution for simple tasks. Cloud services are services that are deployed on a cloud computing structure to take advantage of its computation and distribution qualities. A common use is the data replication service at websites and mobile applications, so even if we have increasing access at some moment, the cloud service can instantiate another machine or machines to provide minimum latency and avoid processing overhead on the servers.

In our work we decided to evaluate the push notification service of Pusher. The main intention here is to provide an easy way to exchange messages between mobile devices around the world with minimum codification effort.

4. PUSHER CLOUD SERVICE

Numerous mobile applications have a focus on fast message delivery to a high number of devices in many parts of the

[1] https://pusher.com/

world. A good example of this service is an email application that sends us a notification whenever we receive a new message without requiring us to actively request new information. This approach is called push notifications, and one implementation of it is Pusher. This cloud service uses cloud computing solutions in order to offer a service that delivers messages through web sockets and HTTP streaming.

One of the advantageous features supported by Pusher API is its support of HTTP Keep-Alive feature. Once connected to Pusher cloud service, it is possible to send and receive messages to/from the cluster without starting a new connection, saving the overhead of restarting new TCP connections. The Pusher service offers the possibility of creating real-time applications considering all of its resources.

Although Pusher has many paid plans with more resources available, we decided to evaluate the free plan and verify if it can be used successfully in specific use cases. The free plan has a limit of 100,000 messages per day and it counts the API requests and messages delivered to each client. We can have up to 20 clients connected at the same time and all clients will send and receive messages only from the US-East cluster server that is situated in Northern Virginia.

The service also has some restrictions for all plans, free and paid alike. Clients are allowed to send no more that 10 messages per second and will be disconnected from the service when they exceed this limit. This limitation is due to the overhead on distributing messages among a thousand users. Every message has a size limit of 10 kilobytes, but it is possible to request an upgrade in order to send larger messages.

The requirement to exchange messages between users is to create a channel and have users connected to the same channel. The API supports public, private, and presence channels. Public channels are used only to send messages from the server to the users connected, like a feed. Private channels need a prefix "private-", require server authentication, and offer the option to accept messages from its users. The presence channel is similar to the private channel, and includes the feature of requesting information about connected users. The channels may have different events that can be bound by clients, e.g. a client can bind the event "client-event" and receive notifications when a new event with the same name is sent to the channel. On private and presence channels, client events must have the "client-" prefix. An example code used to send and receive messages through the Pusher cloud service using a private channel is presented on Listing 1.

In this piece of code we have omitted some information that depends on the application. The first is the "AUTH_PAGE", that needs to be configured to give a unique authentication code to every socket connection. The "PUSHER_API_KEY" is the key received when creating an application on Pusher. After creating the channel, we need to bind an event with an event listener. The event listener requires the code that is expected to run every time the event occurs. The "MESSAGE" is a string with any values defined to be sent, e.g. numbers need to be converted directly to string or through Base64 binary-to-text encoding schema.

Pusher can send any type of data through the cloud respecting plan limits. For instance, one might share codes from computer music languages like CSound [2], ChucK [3], and SuperCollider [4]. Cloud services can also share symbolic data or control signals between any mobile applications, allowing one to make use of any computer music synthesis engines available in tandem with Cloud services. On the next section we are going to present the evaluation proposed to verify the utility of push notifications on computer music in a large scale context.

```
Pusher pusher;
PrivateChannel channel;

HttpAuthorizer authorizer =
        new HttpAuthorizer(AUTH_PAGE);
PusherOptions options = new PusherOptions();
options.setAuthorizer(authorizer);
pusher = new Pusher(PUSHER_API_KEY, options);
pusher.connect();

String channelName = "private-channel";
channel = pusher.subscribePrivate(channelName);
String eventName = "client-event";
channel.bind(eventName,
        new PrivateChannelEventListener() {...});
JSONObject jsonObject = new JSONObject();
String message = MESSAGE;
jsonObject.put("message", message);
channel.trigger(eventName, jsonObject.toString());
```

Listing 1: Example of Java code from Pusher API

5. EVALUATION

Our core metric for evaluation of push-notification-based services in large scale computer music application is round-trip time (RTT). The *Pusher.com* cloud service has been chosen due to its popularity and hard limits on the free plan. For comparison, another available option would be the PubNub [5] service, which at this time offers a free plan that has a limit of one million messages per month and messages up to 2KB.

Some pilot tests performed before the full evaluation presented an intermittent loss of connection when exactly 10 messages were sent per second, which represent the nominal maximum allowed. To avoid this issue a 150ms delay between messages was used, and we established 10 cycles of 100 messages per test with a delay of 500ms between each cycle during the tests. Each test represented a performance of approximately 3 minutes with these definitions.

Six tests were defined based on the number of floats sent as arguments inside the messages. We selected to evaluate messages with 1, 50, 100, 150, 200, and 250 floats. In this case, the evaluation simulated messages in a range from 7 to 1750 floats per second. The messages had five divisions: a unique device id; message number with cycle number on the range of thousands; the total number of messages; a random integer as a key for message identification; and a block of floats.

[2] CSound: http://www.csounds.com/
[3] ChucK: http://chuck.cs.princeton.edu/
[4] SuperCollider: http://supercollider.github.io/
[5] PubNub: http://www.pubnub.com/

An example of a message used during the tests is presented below:

23.0.1.A87422113 1001 1000 76 [0.28506452]

We created an Android application based on the loop-back concept, so one device would send messages, the "*sender*", while the other device, the "*loopback*", would need to answer the message as soon as possible, simulating a loop-back circuit. The first device registered all messages sent and received on a report file with millisecond time-stamp precision. The method used to register the events on the report was *SystemClock.elapsedRealtime()*. We also run AsyncTask invoking *executeOnExecutor()* with *THREAD_POOL_EXECUTOR* in order to have parallel execution on Android. An activity diagram is presented on Figure 1. The *loopback* device also registered sent messages on report with the same precision.

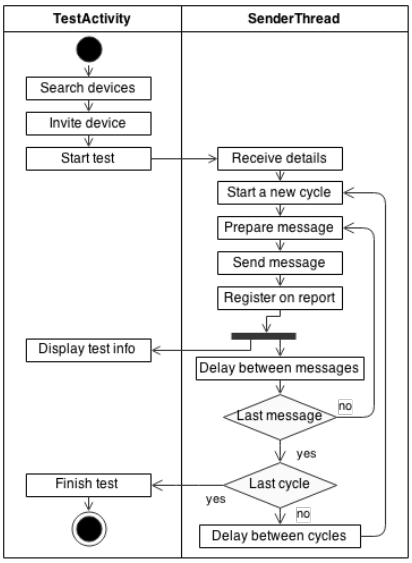


Figure 1: Activity diagram of the test from the sender point of view.

Two mobile devices were used for this evaluation: LG D685 G Pro Dual Lite (D685) and Sony Xperia Z3 Compact (Z3) [6]. The tests were performed between three different universities on the American continent. The mobile device D685 was selected to be the *loopback* device at the Federal University of Paraíba (João Pessoa, PB, Northeastern Brazil). On the other hand, the Z3 was defined as *sender* from two different universities. The first evaluation was performed from the University of São Paulo (São Paulo, SP, Southeastern Brazil), and the second from the University of Michigan (Ann Arbor, MI, USA). We decided to use the Z3 as the sender due to its quad core processor that could reduce the problems with system latency.

The routes between the universities and the cluster used by the cloud service are presented on Figure 2. It is important to notice that every message needs to be sent to the cluster before being pushed to its final destination. This means that every message exchanged between São Paulo and João Pessoa passes twice through São Paulo before going back to São Paulo.

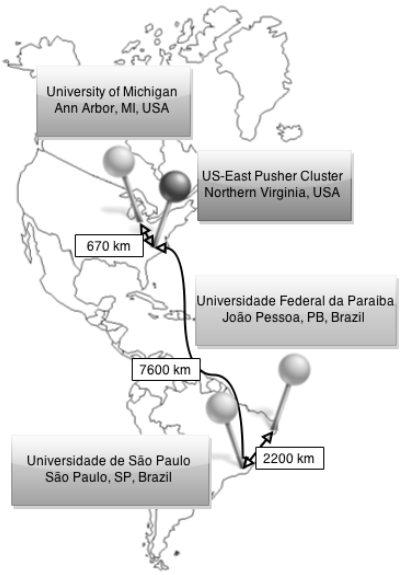


Figure 2: Routes between the universities with linear distance in kilometers

6. RESULTS

The geographic configuration imposes physical restrictions on the lower bounds of networked performance. We made use of the broadband Internet connection between the universities, whose actual medium of interconnection is optical fiber, and the speed of light in this medium is about 200,000 km/s. If we consider the distances on the map at Figure 2, we have the light RTT as 174ms on the first evaluation, between São Paulo and João Pessoa (SAO-JPA) [7], and 104ms on the second evaluation, between Ann Arbor and João Pessoa (ARB-JPA) [8].

The RTT for each message sent is presented in the charts of Figure 3. Some values overshoot the chart limit to maintain the scale of all charts. The results that reach the ground

[6] Comparison between the devices: `http://www.gsmarena.com/compare.php3?idPhone1=5736&idPhone2=6538`

[7] The route between São Paulo and João Pessoa is 34,800km.

[8] The route between Ann Arbor and João Pessoa is 20,940km.

represent a loss of message or connection during evaluation. Table 1 and Table 2 show a summary with the main results extracted from these measurements.

Floats	1	50	100	150	200	250
Lost msgs	14	26	25	3	21	38
Msg size	41	614	1190	1782	2355	2950
Minimum	342	332	332	329	332	352
Maximum	2430	3916	4371	1595	3014	1700
Average	515	578	563	486	536	543
Std. dev.	224	366	394	181	305	168

Table 1: SPA-JPA tests: Results from RTT evaluation using cloud services between São Paulo and João Pessoa. RTT is in milliseconds and average message size is presented in bytes consisting of 1 byte per character.

Floats	1	50	100	150	200	250
Lost msgs	3	0	0	17	5	0
Msg size	43	613	1189	1784	2378	2935
Minimum	166	172	172	182	199	190
Maximum	1953	1052	898	3100	1869	951
Average	243	230	273	316	348	329
Std. dev.	138	83	103	317	143	101

Table 2: ARB-JPA tests: Results from RTT evaluation using cloud services between Ann Arbor and João Pessoa. RTT is in milliseconds and average message size is presented in bytes consisting of 1 byte per character.

We can see in the charts that the RTT is better on ARB-JPA evaluation. Another point from ARB-JPA evaluation is that the more floats we have on a message, the higher RTT we find. The SAO-JPA evaluation presented some instability regarding the average RTT and there was no discernible pattern linking the number of floats to the average RTT.

Although we have had some high RTT values during the tests, the concentration of high RTT values appear to be more frequent in clustered sequential messages than when messages are isolated (meaning musical algorithms based on sporadic communication would suffer less). Furthermore, it was relatively common for the service to lose at least one message after a high RTT value.

The results summarized in the Table 1 present a minimum RTT of 329ms with 150 floats and average RTT values between 486-578ms for all message sizes. On the other hand, we have a minimum RTT of 166ms with 1 float on Table 2, and average RTT values between 230-348ms. We sent 1000 messages on each test and lost no more than 4% of the messages in the worst case, and we lost more messages in the SAO-JPA evaluation than we lost in ARB-JPA. Moreover, ARB-JPA evaluation had tests with zero message loss.

7. DISCUSSION

A common scenario for network music performance expects minimum latency when synchronization is at stake. Participants' locations are an important factor in this case. When participants are in the same place, technologies for local networks might offer the best solutions, whereas if we want participants from different places or continents, the Internet would be the obvious choice. Despite the fact that those are easy choices, the merging of solutions for handling simultaneously both scenarios may not be easy to achieve. Moreover, depending on the technology used, some participants will have difficulties to engage in the performance due to technical restrictions. In our work we decided to evaluate a cloud computing solution for handling both situations at once.

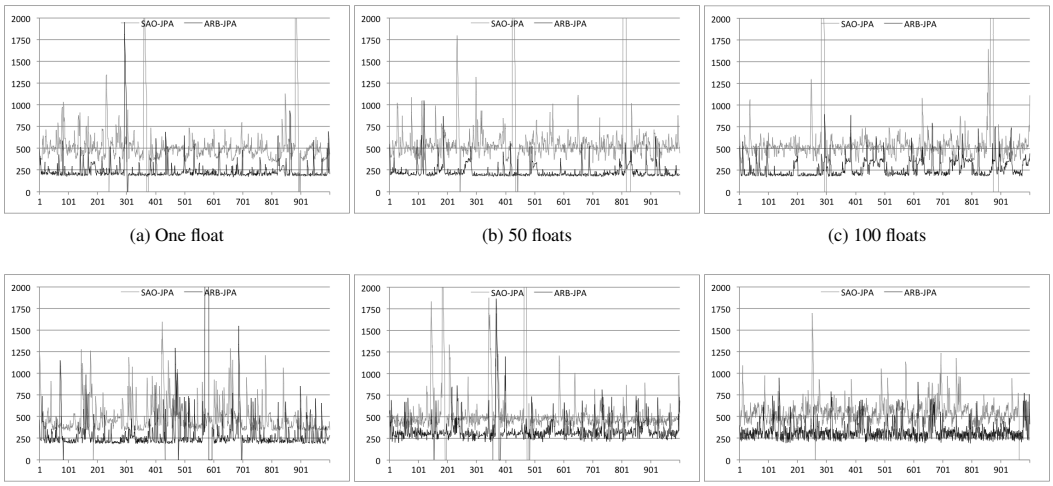
We evaluated the communication between mobile devices from different locations using the Pusher cloud service and WiFi connection. We expected some differences due to the length of the route connecting the cities, the time to process the messages in all endpoints, and delays caused by any network congestion, since those personal devices were not using a dedicated connection. Notwithstanding, we got good results that are going to be discussed on this section.

First of all, the location of the clusters used by the cloud servers might be known or better selected beforehand. During our tests, we found out that we were taking redundant routes as a consequence of the specific cluster for the free plan used on Pusher. The paid plan allow the definition of the cluster, and we recommend a previous verification of the possibilities before signing up to any cloud service.

The advantages of the cloud service offered by Pusher are many: we can exchange messages through a robust cloud computing structure with just a few lines of code; we do not need to care about any other cloud computing configuration (e.g. virtual machines, web server); they have libraries for many languages used on desktop computers and mobile devices, and the same application can interact using both platforms at the same time; and we can also send any kind of data using any string conversion or encoding, for instance, any Base64 encoding schemes.

Although the free plan fitted our evaluations, it has some disadvantages for music performances. The users are restricted to one cluster localization and they need to upgrade to the paid plans, in case a performance requires lots of users and messages. We have reached daily limits with our evaluations when we tried to repeat some tests, and we had to wait 24h to restart everything. Another disadvantage is that apparently we can only use one cluster location per application, but it is possible to transit between clusters under some special conditions (e.g. on presence channels we can't have clients connected during transitioning).

Despite of that, computer music performances can experience new possibilities for collaborative works using this cloud service and taking everything into account. Audience participation is also a good target for users of cloud services due to its easy integration with WebAudio applications that can run on browsers of both mobile devices and computers. In this



(a) One float (b) 50 floats (c) 100 floats

(d) 150 floats (e) 200 floats (f) 250 floats

Figure 3: RTT for each message.

case, the audience will only need an Internet connection to access the application through a compatible browser.

The use of symbolic data reduce the costs of data transmission for participants, and might be better accepted than an audio application that can result in an expensive bill in the end of the month considering the prices of current 4G plans (particularly in Brazil at this point). Moreover, the audio synthesis is becoming less problematic while most of actual mobile devices are multi-core enabled and support any programming language with libraries to mitigate problems in audio processing.

Considering half of RTT as latency, we got a minimum of 83ms and averages between 115 and 289ms in our tests, even with large messages, and we expect that better results can be achieved in places near the clusters. The advantages of using cloud computing in musical performances are also related to reliability in message distribution. We lost just a few messages, less than 4% on each test, and we had many tests without losing any messages.

We also took advantage of HTTP Keep-Alive feature to reduce the overhead of TCP connections. We can infer that cloud computing is a reliable alternative when compared to UDP, which is one of the most used solutions for collaborative musical performances but cannot be easily adapted to multiple mobile devices hidden under many widespread local networks.

Even though these results may suggest that the free plan of this particular cloud service is not suited for a highly synchronized performance across the continent, or a performance highly sensitive to data losses, in other contexts latency and losses may be absorbed in musical settings with user interaction without being perceived as flaws or problems. Some compositions or performances could rely on sounds with long attacks or with very smooth variations so as to minimize (or even overcome) latency issues during music performance. Another interesting approach to handle latency is to work with two time-lines: one for preparing and composing and another for sharing and performing, so that participants can spend time creating short music sequences on their devices before sending, without worrying about getting it wrong somehow.

8. CONCLUSION

The delay between messages is an important setting on the Pusher cloud service. Message loss may have occurred in our evaluations due to overhead on the network buffer. In the event that at least two packets are grouped by a network buffer at any point before being sent to Pusher, we will probably have more than 10 messages per second even if we decide to use 150ms delay between packets. Pusher will disconnect the user at this moment, and will try to reconnect the socket as soon as possible. Although some messages might be lost in this situation, the application developer can bind the disconnection event and stop sending messages until the device gets connected again.

Paid plans offer different clusters that can be selected depending on routes between performers or application users. Although we have lots of clusters available around the world on Pusher cloud service, it may be necessary to try another cloud service if the localization of these clusters are not suitable. At the time of writing this paper, Pusher still does not have clusters in Brazil, and we would suggest other cloud services like PubNub in the case that all participants are from this country.

We have showed that the computer music community may use cloud services as a new way of intercommunication in musical applications, a way that facilitates implementation and improves scalabity of previous musical ideas that were conceived for small groups of participants. The delay and reliability of the service has proven to be suitable for many applications and opportunities, and we may expect improvements in a near future. Push-notifications help users to send and receive data from any part of the world and control the final sound from their own mobile devices, using other users' inputs to influence their music creation in a collaborative way. We expect that many musical works aimed at local networks may now be extended to cloud-based services.

Acknowledgments

This research would not be possible without the support of CAPES [9], RNP [10], Computer Music Research Group [11] and NuSom Research Centre on Sonology [12] at USP, the Digital Video Applications Lab (LAViD) [13] at UFPB, the mobile music group at the University of Michigan, and Internet2 [14].

9. REFERENCES

[1] C. Bartlette, D. Headlam, M. Bocko, and G. Velikic, "Effect of Network Latency on Interactive Musical Performance," *Music Perception: An Interdisciplinary Journal*, vol. 24, no. 1, pp. 49–62, 2006.

[2] C. Chafe and M. Gurevich, "Network Time Delay and Ensemble Accuracy: Effects of Latency, Asymmetry," in *Audio Engineering Society Convention 117*, Oct 2004. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=12865

[3] A. Harker, A. Atmadjaja, J. Bagust, and A. Field, "Worldscape laptop orchestra: Creating live, interactive digital music for an ensemble of fifty performers," in *International Computer Music Conference*, 2008.

[4] J.-P. Cceres and C. Chafe, "JackTrip: Under the Hood of an Engine for Network Audio," *Journal of New Music Research*, vol. 39, pp. 183–187, 2010.

[5] J. W. Kim and G. Essl, "Concepts and Practical Considerations of Platform-Independent Design of Mobile Music Environments," in *International Computer Music Conference*, 2011, pp. 726–729.

[6] A. J. Bianchi and M. Queiroz, "On the performance of real-time DSP on Android devices," in *Sound and Music Computing Conference*, 2012, pp. 113–120.

[7] G. Weinberg, "The aesthetics, history, and future challenges of interconnected music networks," in *International Computer Music Conference*, 2002, pp. 349–356.

[8] G. Levin, "Dialtones - A telesymphony, September 2001," Brucknerhaus Auditorium, Linz, Austria, 2001. [Online]. Available: http://www.flong.com/projects/telesymphony

[9] G. Geiger, "PDa: Real time signal processing and sound generation on handheld devices," in *International Computer Music Conference*, 2003.

[10] M. Rohs, G. Essl, and M. Roth, "CaMus: live music performance using camera phones and visual grid tracking," in *New Interfaces for Musical Expression*, 2006, pp. 31–36.

[11] G. Essl and M. Rohs, "ShaMus: A sensor-based integrated mobile phone instrument," in *International Computer Music Conference*, 2007, pp. 200–203.

[12] A. D. de Carvalho Junior, "Sensors2PD: Mobile sensors and WiFi information as input for Pure Data," in *Joint Conference: 40th International Computer Music Conference and 11th Sound and Music Computing Conference*, 2014.

[13] G. Wang, G. Essl, and H. Penttinen, "Do mobile phones dream of electric orchestras," in *International Computer Music Conference*, 2008.

[14] I. I. Bukvic, "A Behind-the-Scenes Peek at World's First Linux-Based Laptop Orchestra The Design of L2Ork Infrastructure and Lessons Learned," in *Linux Audio Conference*, 2012.

[15] H. Choi and J. Berger, "Waax: Web audio api extension," in *New Interfaces for Musical Expression*, 2013, pp. 499–502.

[16] C. Roberts, G. Wakefield, and M. Wright, "The Web Browser as Synthesizer and Interface," in *New Interfaces for Musical Expression*, 2013, pp. 313–318.

[17] F. L. Schiavoni, M. Queiroz, and M. Wanderley, "Alternatives in network transport protocols for audio streaming applications," in *International Computer Music Conference*, 2013, pp. 193–200.

[18] N. Weitzner, J. Freeman, S. Garrett, and Y.-L. Chen, "massMobile - an Audience Participation Framework," in *New Interfaces for Musical Expression*, 2012.

[19] J. Allison, Y. Oh, and B. Taylor, "NEXUS: Collaborative Performance for the Masses, Handling Instrument Interface Distribution through the Web," in *New Interfaces for Musical Expression*, 2013, pp. 1–6.

[9] CAPES: `www.capes.gov.br`
[10] RNP: `www.rnp.br`
[11] Computer Music Research Group: `compmus.ime.usp.br`
[12] NuSom Research Centre on Sonology: `www.eca.usp.br/nusom`
[13] LAViD: `www.lavid.ufpb.br`
[14] Internet2: `www.internet2.edu`

[20] A. Hindle, "SWARMED: Captive Portals, Mobile Devices, and Audience Participation in Multi-User Music Performance," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2013, pp. 174–179.

[21] ——, "CloudOrch: A Portable SoundCard in the Cloud," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, B. Caramiaux, K. Tahiroglu, R. Fiebrink, and A. Tanaka, Eds., London, United Kingdom, Jun. 2014, pp. 277–280.

F.8 ICLC 2015 - SuperCopair: Collaborative Live Coding on SuperCollider through the cloud

Paper details

Title: *SuperCopair: Collaborative Live Coding on SuperCollider through the cloud*
Authors: Antonio Deusany de Carvalho Junior, Sang Won Lee, Georg Essl

Conference details

Title: International Conference on Live Coding (ICLC)
Venue: School of Music, University of Leeds, UK
Dates: July 13 to 15, 2015

SuperCopair: Collaborative Live Coding on SuperCollider through the cloud

Antonio Deusany de Carvalho Junior
Universidade de São Paulo
dj@ime.usp.br

Sang Won Lee
University of Michigan
snaglee@umich.edu

Georg Essl
University of Michigan
gessl@umich.edu

ABSTRACT

In this work we present the SuperCopair package, which is a new way to integrate cloud computing into a collaborative live coding scenario with minimum efforts in the setup. This package, created in Coffee Script for Atom.io, is developed to interact with SuperCollider and provide opportunities for the crowd of online live coders to collaborate remotely on distributed performances. Additionally, the package provides the advantages of cloud services offered by Pusher. Users can share code and evaluate lines or selected portions of code on computers connected to the same session, either at the same place and/or remotely. The package can be used for remote performances or rehearsal purposes with just an Internet connection to share code and sounds. In addition, users can take advantage of code sharing to teach SuperCollider online or fix bugs in the algorithm.

1. Introduction

Playing in a live coding ensemble often invites the utilization of network capability. Exchanging data over the network facilitates collaboration by supporting communication, code sharing, and clock synchronization among musicians. These kinds of functions require live coding musicians to develop additional extensions to their live coding environments. Due to the diversity of the live coding environment and the collaboration strategies settled for performances, implementing such a function has been tailored to meet some ensemble's requirements. In addition, networking among machines often requires additional configuration and setup, for example, connecting to specific machines using an IP address. In order to overcome these constraints, our goal is to realize a platform that facilitates the collaboration among live coders with minimal efforts of configuration, utilizing cloud computing.

There are many advantages to replacing a traditional server-client system with a cloud server. First of all, the collaboration scenario could be extended to the live coding ensemble whose members are distributed over different locations, enabling a networked live coding performance. Not only does this enable telematic performances, but it will also make a live coding session take place in a distributed manner, which will change the rehearsal process of live coding ensembles, whether it is remote or co-located. In addition, using the cloud server minimizes the amount of setup needed for networking as long as each computer is connected to the Internet. The amount of setup required is equivalent to creating a shared document in a Google Drive.

To that end, we present **SuperCopair**, a package for the Atom text editor, that offers code sharing and remote execution over the Internet. In this paper, we introduce the background that the idea is built upon, articulate our motivations for using cloud computing, and describe the implementation of the system. Finally, we suggest multitudes of new performance practices enabled by the system.

2. Networked collaborative live coding

Networked collaboration in live coding was present from the inception of live coding where multiple machines are clock-synchronized exchanging TCP/IP network messages (Collins et al. 2003). Many live coding ensembles also utilize network capability to share data and communicate within the ensemble (Collins et al. 2003; Rohrhuber et al. 2007; Brown and Sorensen 2007; Wilson et al. 2014; Ogborn 2014a). However, most of the time, they are based on the local network communication and not designed for remote collaboration attempted in the tradition of Network Music. Remotely connected music systems not only create a number of unique challenges and aesthetic opportunity as a performance in public, but also provide a base for musicians in different localizations to collaborate over the network synchronously.

Telepresence performance recently emerged as a new collaboration practice in live coding. Swift, Gardner, and Sorensen (2014) conducted networked performance between two live coders located in Germany and United States using an SSH server located in Australia. Extramuros, a language-neutral shared-buffer, is a web-browser based system to share code among connected machines (Ogborn 2014b). Gibber, a live coding environment on a web browser, supports collaborative editing and remote execution similar to Google Docs (Roberts and Kuchera-Morin 2012). Commodity softwares (such as Google Docs, CollabEdit, or DropBox) can be useful for remote collaboration and are convenient since users do not need to perform any configuration. However, these systems were either not designed or offer at best limited support for remote music performances.

3. Designing a collaborative solution

Although in the past it was difficult to think of thousands of people interacting at the same time on a musical system, the actual situation is in the quest of the best way to use the services and technologies offered day after day. For the last several years, we have witnessed an impressive advancement in the quality of services of cloud computing systems. Cloud servers distributed worldwide are connected through fiber optic broadband, and its cloud services have many advantages for computer music and collaborative works. We are taking some benefits from these characteristics in this study.

The cloud computing suggests itself as the next logical step in network capability, ready to be used for musical applications and performances. 'CloudOrch' is one of the first attempts to utilize the advantages of cloud computing in musical ways (Hindle 2014). The idea was to deploy virtual machines for client and server users, create websockets for intercommunication, and stream audio from cloud instruments to both desktop computers and mobile devices using web browsers. The author dubbed his idea 'a sound card in the cloud' and presented latency results from 100 to 200~ms between the Cybera cloud and the University of Alberta, Canada using the HTTP protocol. Using cloud computing as opposed to using server-client option has multiple advantages. Once an online instance is configured, a user can connect to or disconnect from the cloud at any time and can share the same resources within a group of connected users and take advantage of the reliable and scalable resources provided. Indeed, this solution can be useful for live coding network music.

The authors had already discussed models and opportunities for networked live coding on a past work (Lee and Essl 2014). The paper introduces diverse approaches in networked collaboration in live coding in terms of the type of data shared: code sharing, clock synchronization, chat communication, shared control and serializable data (such as audio). We draw upon ideas of existing systems that enable 'code sharing' and 'remote execution' re-rendering program state by evaluating code fragments in both the local machine and the remote machines in many live coding environments and extensions (Brown and Sorensen 2007; Rohruber and Campo 2011; McKinney 2014; Roberts and Kuchera-Morin 2012). These systems are similar in the sense that they need a separate server installed and configured by their users.

It is a general trend to have software application distributed over the Internet. Cloud computing is the central tool to realize the distributed softwares. However, the use of cloud computing is underdeveloped in computer music and we believe that it is the next logical step to put computer music applications in the cloud as a mean to realizing network music. The cloud computing provides a set of services that are beneficial to scale the computer music performance. For example, we can imagine a small scale ensemble co-located in the performance space, in which case the cloud computing will create a virtual machine based on the data center nearby the performance location. In the opposite case where large-scale participants are expected on a collaboration session, the cloud service will easily scale its computational power, network traffic bandwidth and storage space automatically to meet the spontaneous needs, although it will have some monetary cost.

In terms of network latency, we have achieved, an average round-trip time of 230~ms between Brazil and United States, and a minimum of 166~ms (Carvalho Junior, Queiroz, and Essl 2015). These tests were done using mobile devices connected to Pusher, a cloud service described below, but it can be extended to almost any device connected to the Internet. The strategy of transferring code (textual data) and re-rendering the program state remotely instead of streaming audio makes the latency less critical particularly for the scenario of live coding. However, it should be noted that the sound outcome from local machines and remote machines may not have exactly the same sound for many reasons (e.g., latency, randomness, asynchronous clock, packet loss).

The use of cloud computing resources became easier after the introduction of some cloud services that create an abstraction of the cloud computing set up and offer simple APIs for users, as we can find on Pusher. Pusher offers a cloud computing service that delivers messages through web sockets and HTTP streaming, and support of the HTTP Keep-Alive feature. The service has a free plan with some daily limitations such as 100,000 messages and a maximum of 20 different clients connected. Another limitation of the free plan is that we can only exchange messages through the US-East cluster server situated in Northern Virginia. The paid plans are more flexible and they make possible to have more users connected, send more messages, and use other clusters. In spite of that flexibility, all plans have a hard limit of 10

messages per second for each user. This limitation is due to the overhead of message distribution among a thousand users, but it really suits most needs to common use cases. Every message has a size limit of 10 kilobytes, but one can request an upgrade if larger messages are needed. Although it has limitations, we do not need to set up any cloud instance to benefit from the cloud computing infrastructure provided by this service.

The service works with push notifications, so every message sent is going to be received by all devices assigned to the same channel. A SuperCollider programmer can evaluate the whole code, a selected part, or just a line using keyboard shortcuts. SuperCollider programming language supports real time audio synthesis and is used extensively by live coders. These characteristics turn the language very suitable to be used with a push notification cloud service.

4. SuperCopair

The solution presented in this paper was created as a package to the Atom.io IDE. Defined as 'a hackable text editor for the 21st Century' on its site[1], Atom is a text editor created using web technologies and has its development powered by the github community. This IDE has numerous packages for many programming languages and presents some solutions for coding, debugging, and managing projects. Atom packages are programmed in CoffeeScript, which is a programming language that can easily be converted to Javascript and can also integrate its libraries. The developers can install Atom packages to enable various functionalities in the IDE such as: communicate through chats, use auto-complete in certain programming language syntax, interact with desktop and web applications, integrate with the terminal command line, and have many options based on other packages. These features have motivated the development of SuperCopair package for Atom.

SuperCopair is based on two Atom packages: atom-supercollider and atom-pair. The first package turns Atom.io as an alternative SuperCollider IDE and permits users to openly communicate locally with SuperCollider audio server through OSC in the same way we can do on SC-IDE. Moreover, the users can take advantage of other Atom packages additionally to quarks packages. The latter package is used for pair programing through the Internet. The atom-pair package is based on Pusher cloud service and its default configuration is based on the community free plan, but a user can modify the settings and use the user's own keys within the personal free or paid plan.We decided to merge both packages to add new features for collaborative live coding, and finally had dubbed it the SuperCopair package.

The main idea is that all participants have the opportunity to evolved into a collaborative performance.

The IDEs for SuperCollider have, by default, shortcuts to evaluate a line, a block, and to stop all sound process that is running. In addition to these options, the SuperCopair package includes methods and shortcuts to broadcast these events and execute them on all users connected at the same pairing session. Through the shortcuts, one can decide to evaluate selected code either only in the local machine, or in all computers of the session. One can also turn on and off a broadcast alert option in the settings in order to be asked or not before evaluating every broadcast event sent by another user in the same session. This allows each individual has control over which code to be evaluated in the local machine.

The broadcast events are diffused through the cloud service and are expected to be evaluated as soon as each device receives the event message. The message includes the code to be evaluated and the user identification. A representation of a session using SuperCopair package is shown at Figure 1.

4.1. Package installation and use

One can have the package properly installed and ready to be used in two different ways. Via the Settings View in Atom.io, the user can search and install the package. It is also possible to install the shell commands during Atom.io setup and use the line below to install the package:

```
apm install supercopair
```

After the installation, the user needs to start a new session before inviting others. An instructional step-by-step setup is presented on the package page. Then one can get initiate a performance by opening a SuperCollider file and starting a pairing session. The session ID string needs to be shared with collaborators so they can use the string to join the same session.

The shared session ID is based on the channel created at the cloud service and it contains application keys. It is recommended to change the app keys after each session. As the keys are linked to the account used during the performance, other participants can use the keys for other activities and the number of events will deducted from the main account.

[1] Atom.io website: http://atom.io/

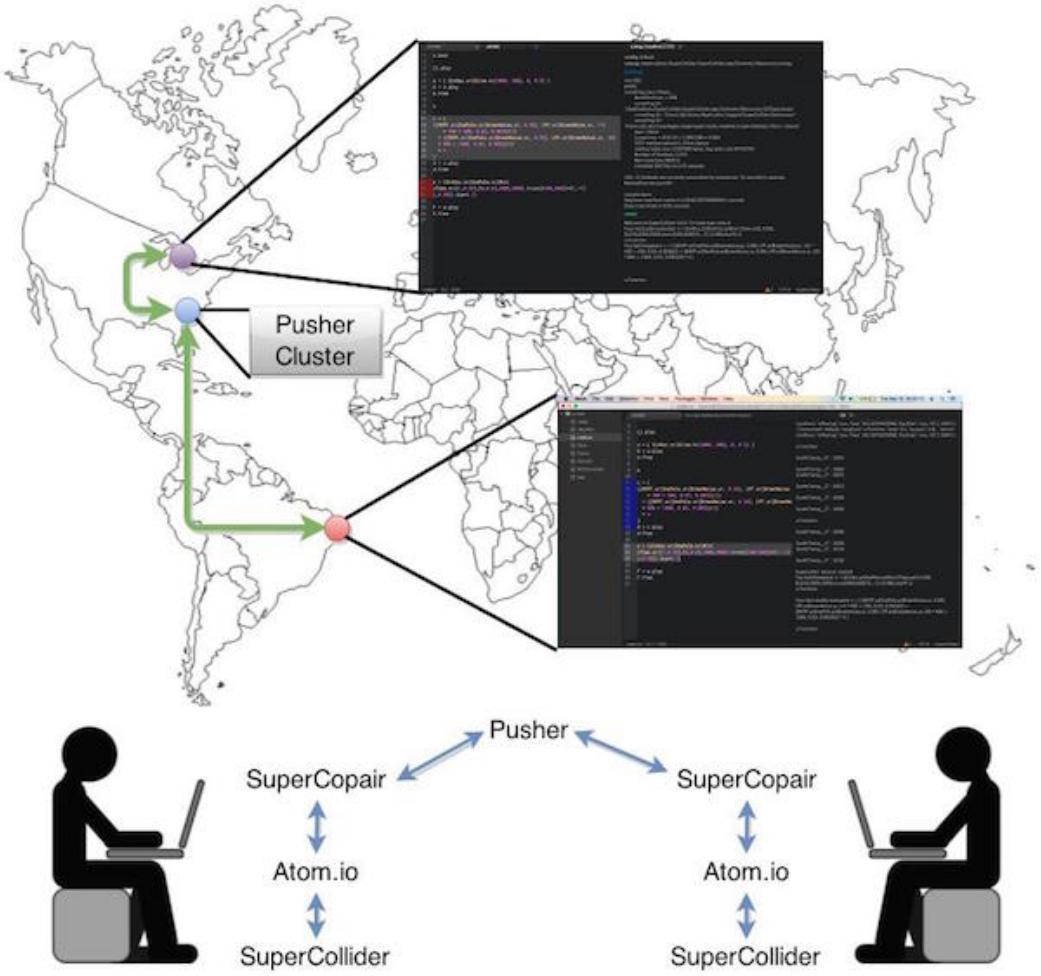


Figure 1: *Example of a session using SuperCopair and the architecture of interactions. The central bullet is the localization of the cluster server in Northern Virginia, and the other bullets represents users connected to the cloud server. The screen has the code on the left and SuperCollider post window on the right. The architecture presents only two users but it can be replicated to many, with diffusion on Pusher cloud service.*

The users who joins later will see the most recent version of the shared code. The users are identified by different color markers, and they can identify what each member is writing on the file based in these colors. A pop up provides information about users joining or leaving the session. Furthermore, a message including the identification of the user and also the code evaluated is shown at SuperCollider post window right after each broadcast event is evaluated. In case the broadcast alert option is on, a dialog will appear whenever an event message is received from another and ask if the user would accept or reject the code evaluation. The alert dialog will have the sender's id and the code sent via broadcast. When a live coder leaves the session, he or she can keep the most recent updated file to save or edit offline.

The delay achieved on the free plan depends on the distance between every member and the US East Coast cloud server. This free plan from Pusher cloud service allow 20 different clients per day on the same session and 100 thousand messages per day, however we have higher limits on paid plans. The session will stop after reaching the daily limit of messages for all plans, but the daily quota is reset after midnight UTC. It is important to keep these details in mind while facing any problem or high latency. The user may want to try a paid plan to have distributed data center, clients more than 20, and larger sized messages.

4.1.1. Shortcuts

The users have some special shortcuts depending on the operating system, and they are related to these specific functions:

- Compile library (open post window if needed)
- Clear post window
- Evaluate selection or current line locally
- Panic ! Stop all music
- Broadcast a code evaluation to everyone (including oneself) in the session
- Broadcast a code evaluation to others (excluding oneself).
- Broadcast stop command to everyone (including oneself) in the session
- Broadcast stop command to others (excluding oneself).

These shortcuts can be used to interact with other participants during a performance. The broadcast methods will only be shared with users on the same session, so it is also possible to create multiple sessions and interact with different crowd teams at the same time using a distinct Atom.io window on the same computer.

4.1.2. Practices and performances

The authors attempted to test the application multiple times in the co-located setup and also tried remote sessions by recruiting SuperCollider users. From one of our practices, there were live coders from Ann Arbor, MI, and San Francisco, CA, in U.S., and also São Paulo, SP, and Fortaleza, CE, in Brazil. During the session, participants (including the author) shared the session ID using an Internet Relay Chat (IRC) channel and we had a brief discussion about the package before starting to code. Some users reported that it could be dangerous to use headphones if we had switched off the alert for broadcast events, because some user may send a louder code to be synthesized. In the end, the session is successfully carried out without many problems and we are on the improvement of the package based on comments and suggestions from the participants. Atom.io installation was cumbersome for some users of Linux due to recompilation requirements at some distributions, and a step-by-step guide is under construction. Additionally, Mac users need the newest versions of the system in order to install Atom.io, but the users can also use a virtual machine with Linux and get rid of this limitation.

The practice addressed above is to simulate a networked live coding performance where multiple remote performers join a SuperCopair session from each one's our location, that may not be the concert space. In the local concert space where the audience is, a laptop connected in the session is placed without a performer on stage. Each performer would evaluate code in broadcast mode so that the computer on the stage will generate the collection of sound that remote live coders make via the Pusher. At this performance, the spectators at the local concert hall may have a wrong impression about the performance in the beginning because there is no one on stage except the laptop. As the audience will watch the video projection of the laptop screen, they will understand the main idea of remote performers live coding locally, from the multiple concurrent edits and some live coders' explanatory comments shown on the editor.

5. Discussion and conclusions

Here we present advantages and opportunities enabled with SuperCopair in network live coding: remote collaboration, telematic performance, and crowd-scale networked performance. One interesting advantage of the application is that it supports remote pair programming. We have witnessed that users can teach the basics of a language each other or help in bug fixing using online pair programming. Beginners can invite someone from anywhere in the world for a pairing session and start coding together on the same file and also synthesize the same code in real time while learning some tricks about live coding. Additionally to the forums and mailing lists, one can invite professionals to help on fixing algorithms for audio synthesis and have another kind of experience like pair or group programming on the same code to come up with a solution collaboratively. This package supports only SuperCollider namespace, but in the near future we can have similar packages for Csound, Chuck, or any other computer music programming language.

SuperCopair also offers novel forms of networked performances based on message streaming. The work of Damião and Schiavoni (2014) is a recent attempt to use network technologies in order to share contents among computers for a musical performance. The authors send objects and strings through UDP using OSC and can control other machines running the same external on Pure Data. They opted for UDP and Multicast to get better results on message distribution if compared to TCP and broadcast, which usually include three way handshaking and relay server. Although their decision has been based on solid arguments, our solution takes advantages of HTTP persistent connections using a single TCP connection to send and receive messages, and we also bring a reliable option for broadcast delivery using cloud services capabilities.

The package presented in this paper can be extended as an alternative for other networked live coding APIs. One can cite the Republic quark package that is used to create synchronized network performances. and the extramuros (Ogborn 2014b), a system for network interaction through sharing buffers of any kind of language. The last solution needs to be configured depending on the language andit does not present any easy way to share control (e.g. stop synthesis on SuperCollider) at the moment. Another constraint of both solutions is the need to create and configure a server on one computer to receive connections from clients, and additionally it would be necessary to open network ports or change firewall settings before starting any interaction with remote users.

SuperCopair realizes accessible configuration of network music performances, utilizing the cloud services. There is no need to configure a server or manage any network setting, e.g. routing, firewall, and port. We expect that even inexperienced users will be able to create a session with lots of people. As long as one can install the Atom editor and the SuperCopair package, the creation and participation at remote performances become an easy sequence of one or two shortcuts. Eventually, SuperCopair will simplify the steps to create a collaborative performance and remote rehearsals, and be used by people without network knowledge.

6. References

Brown, Andrew R, and Andrew C Sorensen. 2007. "Aa-Cell in Practice: an Approach to Musical Live Coding." In *Proceedings of the International Computer Music Conference*, 292–299. International Computer Music Association.

Carvalho Junior, Antonio Deusany de, Marcelo Queiroz, and Georg Essl. 2015. "Computer Music Through the Cloud: Evaluating a Cloud Service for Collaborative Computer Music Applications." In *International Computer Music Conference*.

Collins, Nick, Alex. McLean, Julian. Rohrhuber, and Adrian. Ward. 2003. "Live Coding in Laptop Performance." *Organised Sound* 8 (03): 321–330.

Damião, André, and Flávio Luiz Schiavoni. 2014. "Streaming Objects and Strings." *Live Coding and Collaboration Symposium*.

Hindle, Abram. 2014. "CloudOrch: a Portable SoundCard in the Cloud." In *New Interfaces for Musical Expression*, 277–280.

Lee, Sang Won, and Georg Essl. 2014. "Models and Opportunities for Networked Live Coding." *Live Coding and Collaboration Symposium*.

McKinney, Chad. 2014. "Quick Live Coding Collaboration in the Web Browser." In *Proceedings of New Interfaces for Musical Expression (NIME)*. London, United Kingdom.

Ogborn, David. 2014a. "Live Coding in a Scalable, Participatory Laptop Orchestra." *Computer Music Journal* 38 (1): 17–30.

———. 2014b. "Extramuros." https://github.com/d0kt0r0/extramuros.

Roberts, C., and J.A. Kuchera-Morin. 2012. "Gibber: Live Coding Audio in the Browser." In *Proceedings of the International Computer Music Conference (ICMC)*. Ljubljana, Slovenia.

Rohrhuber, Julian, Alberto de Campo, Renate Wieser, Jan-Kees van Kampen, Echo Ho, and Hannes Hölzl. 2007. "Purloined Letters and Distributed Persons." In *Music in the Global Village Conference (Budapest)*.

Rohruber, J., and A. de Campo. 2011. "The Republic Quark." https://github.com/supercollider-quarks/Republic.

Swift, Ben, Henry Gardner, and Andrew Sorensen. 2014. "Networked Livecoding at VL/HCC 2013." In *Visual Languages and Human-Centric Computing (VL/HCC), 2014 IEEE Symposium on*, 221–222. IEEE.

Wilson, Scott, Norah Lorway, Rosalyn Coull, Konstantinos Vasilakos, and Tim Moyers. 2014. "Free as in BEER: Some Explorations into Structured Improvisation Using Networked Live-Coding Systems." *Computer Music Journal* 38 (1): 54–64.

F.9 CLEI 2015 - Cooperative Live Coding as an instructional model

Paper details

Title: *Cooperative Live Coding as an instructional model*
Authors: Antonio Deusany de Carvalho Junior

Conference details

Title: XLI Conferencia Latinoamericana en Informática (CLEI)
Venue: Arequipa, Peru
Dates: October 19 to 23, 2015

2015 XLI Latin American Computing Conference (CLEI)

Cooperative Live Coding as an instructional model

Antonio Deusany de Carvalho Junior
Instituto de Matemática e Estatística
Universidade de São Paulo
dj@ime.usp.br

***Abstract*—The advances on technologies have provided many tools that inspired new instructional models. Learners and instructors are experiencing a diverse environment where everyone can participate from anywhere in the world and share the same learning platforms. Although we already have some manuals, tutorials, and also MOOCs that can be useful for people who wants to learn Computer Music languages, the musical interaction is not offered in these solutions. In this paper we present an instructional model for computer music and live coding based on a cooperative live coding environment where participants can teach and learn through distributed pair programming. We also discuss the fundamental ideas and the tool used on this work during the first experiments.**

I. Introduction

New instructional models have been provided by the advances on technologies and they are becoming popular even without a long evaluation regarding their advantages. Most of them are web based and can be accessed at any time by thousand of students through the Internet. This structure for instructional model takes advantage of the Internet as it is and these models requires a basic setup to some extent. From this point of view, the Internet permits the dissemination of hypermedia to the whole society through the use of digital technologies for information and communication [1].

The proposal of an instructional model self-contained on the Internet also rises from the interaction or interactivity aspects behind the model. One can say that the interaction implies the transmission of information in unidirectional point of view, while the interactivity rises from the two-dimensional participation of the learner [2], or that the interactivity idea is to focus on the observed object that is modified from time to time [3]. Furthermore, the interactivity also includes many features that permits participants to modify and intermediate between the information received and the resultant knowledge.

In this work we are going to present a new instructional model for computer music and live coding based on a tool for cooperative live coding that provides an online environment with many ways of interactivity. We are also opening a discussion about cooperation and collaboration practices throughout the use of this tool, initially proposed for collaborative live coding by the authors. Even though cooperation and collaboration terms may be interchangeable used, we will consider that the cooperation does not imply mutual benefit, and that a collaboration assumes contribution of all participants with mutual goals [4].

The next sections include a discussion about the advantages of new instructional models that involve online systems. We will present how users can learn cooperatively and which environments can be used to provide online interaction. The tool created by the authors is also presented, and we will discuss the advantages of its use for teaching computer music languages with musical interaction.

TABLE I. Sites with MOOCs

Name	Website
edX	https://www.edx.org
Coursera	https://www.coursera.org
NovoEd	https://novoed.com
Udacity	urlhttps://www.udacity.com
MiriadaX	https://www.miriadax.net
Futurelearn	http://futurelearn.com
OpenUpEd	http://openuped.eu
P2PU	https://p2pu.org
UoPeople	http://uopeople.edu

II. Online instructional models

An instructional model is a set of instructions or directions that provide a way to acquire some knowledge, improve capabilities, or extend the practice. Although we have a variety of instructional models available for classroom, in the past two decades many technologies became available and inspired new instructional models.

A new contemporary instructional model is the Massive Open Online Courses (MOOCs). This model offer online courses that can be attended by thousand of people at the same time. The systems often offer syllabus, videos, and forum as the main features for their participants. The idea of MOOCs has been a new tendency in many areas of teaching. People around the world are engaging in online courses alone or in groups and learning anything at anytime. The instructors provide updated materials through the syllabus and propose exercises with online evaluation. The evaluation is automatic in most of the time due to the unlimited number of students that can sign up for the courses. We have many examples of sites with MOOCs in Table I.

The interaction between participants (professors and students) in these sites happens in the forums where anyone can post, answer, and discuss questions at any time. Although the students have this open channel with the professor, the feedback is not always immediate and the professors may take some time to answer questions from all students. It is also noticeable that some instructors have a Teaching Assistant (TA) responsible for the forums. Although the addition of a TA may increase the distance between the students and the instructor, the TA will be more attentive to the forums while the instructor

978-1-4673-9143-6/15/$31.00

TABLE II. MOOCS WITHOUT SCHEDULE

Name	Website
Khan Academy	https://www.khanacademy.org
Udemy	urlhttps://www.udemy.com
Pluralsight	http://www.pluralsight.com
Code School	https://www.codeschool.com
Digital Tutors	http://www.digitaltutors.com
Three House	https://teamtreehouse.com
Veduca	http://www.veduca.com.br
Acamica	https://www.acamica.com
Codecademy	http://www.codecademy.com

will be responsible only to organize the learning objects and syllabus.

The MOOCs are very helpful but some courses are not offered all the time. In case someone has an urgency in learning a specific topic, the MOOCs presented on Table I may not be the best option as these sites present courses on a predefined schedule. However, autodidact (self-taught or self-learner) students have other options of MOOCs sites where the courses are always open. A list of these sites is presented on Table II. In this case the instructors keep the courses open and update the materials from time to time.

The MOOCs have full courses and predefined materials with fixed structure. This instructional model follows the traditional teaching method applied at the classrooms where the student will need to learn everything without any adjustment for his/her own difficulties. While some students will abandon the course due to the lack of basic concepts, other students may find the course tired when they already have previous background and the course does not go further than expected.

In terms of specific content about any topic or special necessities (e.g. learning step-by-step with variable intensity and extra content), the MOOCs may not be the best option. Students interested in technological topics, like programming languages, would probably try online tutorials or specific forums.

Instructors and students are sometimes interested in teaching or learning specific topics that don't need a full course, and the online tutorials are an alternative that can suit better their needs. The tutorials are distributed online mostly in textual formats and can explain the same topic in many levels.

Online tutorials about many subjects and areas can be found through online search tools. Still, the technological tutorials are probably the most distributed through Internet users. The official sites of many programming languages and applications have tutorial sections on their website or include the tutorials inside contents downloaded by users. A list of sites with tutorials is presented on Table III, and these sites may present video tutorials and also courses based on their tutorials.

The W3Schools is one of the most famous site with tutorials about web technologies. All tutorials are deeply detailed including the description of the programming language paradigms, recommendations, attributes, statements, options, and advantages. Additionally, this site also introduces some tools to run code online and execute data base statements through the web interface, that we are going to talk afterward.

TABLE III. SITES WITH TUTORIALS

Name	Website
W3Schools	http://www.w3schools.com
Vogella	http://www.vogella.com
Tutorials Point	http://www.tutorialspoint.com
Tuts+	http://tutsplus.com
TechTutorials	http://www.techtutorials.net
Home & Learn	http://www.homeandlearn.co.uk

Some web technologies has great communities that build online tools in order to help new learners. An example is the Try Ruby[1] website. This is the recommended online tutorial for everyone interested in learning how to program in Ruby language. The site tries to talk with the users like a real instructor and go step-by-step covering the basics of the language. Try Ruby is a short tutorial but it demonstrates how a computer system can interact with a learner during its initial practices. Next we will discuss some solutions where real users (the learners and professionals) can work together in a collaborative and cooperative way.

A. *Cooperative learning*

The use of sites with focus on question and answer (Q&A) extends the instructional models supported by online environments. Although the users may need at least a basic knowledge beforehand, Q&A sites provide search tools for questions related to a specific topic that the users want to learn or to discuss. Some discussions can grow up to many answers and diverse point of view from users around the world. While some users take advantage of these sites only to solve personal problems, many others have the position of supporting other users and expending some time in more detailed discussions and explanations.

The StackOverflow[2] is one example of these sites and it does accept questions about many programming languages. This site is part of StackExchange[3] community which includes other kind of Q&A sites from diverse topics including life, arts, culture, recreation, and science. It is also possible to propose ideas for new Q&A sites at Area 51[4] creation zone from StackExchange.

The StackOverflow site provides some rules that guide initial users in order to help other users to cooperate in solving some technological questions. For programming languages, one of the main rule is to present a minimal, complete, and verifiable example (MCVE). The MCVE will help any other online user to have a detailed idea of the context of the question on most situations. In case the user wants to fix some code regarding Javascript, HTML, or CSS, the recommended procedure is to create a MCVE at JSFiddle[5]. If the user wants to write a specific database statement using SQL, one option to write the MCVE is the SQLFiddle[6]. Both options implement the same idea presented at W3Schools and permits the users

[1]Try Ruby: http://tryruby.org

[2]StackOverflow: http://stackoverflow.com

[3]StackExchange: http://stackexchange.com

[4]Area 51 - The Stack Exchange Network staging zone: http://area51.stackexchange.com

[5]JSFiddle: https://jsfiddle.net

[6]SQLFiddle: http://sqlfiddle.com

TABLE IV. TOOLS FOR DPP

Name	Website
collabedit	http://collabedit.com
CodeShare	http://www.codeshare.io
Cloud9	https://c9.io
Squad	https://squadedit.com
Floobits	https://floobits.com
MadEye	https://madeye.io

to modify the code on the site and see the results without any other specific tool or even the necessity of creating a database for the latter option.

A special feature available at JSFiddle is the collaboration option. The user can share the link of the 'fiddle' created at this site and invite other users to work together on the same code. This idea of having many users working on the same code is also known as pair programming and will be discussed below.

B. Pair programming

There are many ways of working and practicing with programming languages in order to apply agile methods. One famous practice is the Pair Programming (PP), that involves two programmers working on the same piece of code at the same time. This practice suggests only two programmers but it is not restricted to this structure. Although PP is most used by developers that work in companies, research shows that PP can be also used as tool for practicing and also teaching programming [5]. The advances on the technologies have permitted this practices being extended to distributed places.

In the same way as JSFiddle, there are many tools that provide an ambient to share code and allow many users to program together through the Internet. This practice is named Distributed Pair Programming (DPP) and considers users on different machines and locations. There are tools available for DPP and some of them also integrate audio, video, and chat features. A list of DPP tools is presented on the Table IV.

The idea of sharing the code and running online with the results being presented to more than one user at the same time has many advantages. Users can apply the PP concepts where one user only observes while the other one is coding. It is possible to have each user programming a different method on the same file in order to finish the program faster and collaboratively. Some companies can also use this environment to evaluate programmers online from different places and observe how the candidate can solve a problem during an interview. Recent research about DPP literature shows that 'there is a strong trend towards the use and research of the empirical effects of DPP in teaching programming' and that 'there is an opportunity to investigate DPP with other types of collaborative programming' [5].

The fundamentals discussed in this section serve as a base to cooperative live coding practices that are going to be discussed in the next sections. We will start presenting the *live coding* that is a musical practice based on writing code lively and evaluating this code without compilation.

III. LIVE CODING

During the 80's and 90's, composers and performers would write piece of codes in some languages like Csound and wait some time to have the result. [6] discuss that after learning Csound from online documentation in 1996, a specific file with tens of thousands interacting instruments took 20 hours to build, and it sounded dreadful in the end [6, p. 81]. During an interview, Judy Klein remember that during the 80's she would start a compilation of a short piece of sound at night and hear the result in the morning, when no errors had occurred [7]. Nowadays even cheap computers like Raspberry Pi[8] can be used as sound processor with new Computer Music languages, and we can say that the practice of live coding is built upon the technological advances that permit high processing of codes and sound synthesis in a matter of milliseconds.

The idea of writing codes lively is similar to the paradigm behind interpreted languages where the user writes line by line and have the results right after the evaluation has been done. One can cite Javascript as an interpreted language on client side, where the code is not compiled before being executed. Ruby and Python are also normally taken as interpreted language, but this condition depends on the compiler implementation and may differ between versions.

In Computer Music, we have lots of programming languages and some of them are interpreted languages that can be used for live coding. The performers can write pieces of code and hear the results while they continue to write new lines or to modify the same code. No deep programming skills are required for most of the languages used by live coders and the practice have been diffused between people from many areas, specially musicians and artists.

The practice of live coding in music is mostly used on laptop performances and can be achieved with many languages [7]. Research shows that the use of interpreted languages resulted into the rising of live coding movement and that it has become a common practice in festival calls that include electronic music [8]. This art specialization is basically a variation of improvisation and composition using algorithms, and can be briefly defined as 'a form of musical performance that involves the real-time composition of music by means of writing code' [9].

Live coding can also include technical mistakes as any other musical practice [10]. An error on the code can result in an unexpected sound and can generate an undesirable noise or an undesirable traditional pattern depending on the music style. The system can also crash during a performance, and other software or hardware can intercept the sound processing without any control from the performer side. The practice and comfortableness with the language used can give confidence to the performer, and some instructional exercises for this aim are presented on the literature [10].

The live coding practice can also be done by many users at the same time. In this case, the collaboration during a musical performance has a structure similar to a traditional orchestra or band, but the coders may not have a specific

[7]Interview with Judy Klein: http://ias.umn.edu/2013/03/01/electronic-music/#Klein

[8]Raspberry Pi: https://www.raspberrypi.org

instrument, sound, timbre, or function. The performers can also interconnect themselves using some kind of network and share sound or codes, what we would call network music.

Experiments with network music dates back to the 1970's with The League of Automatic Music Composers [11]. The setup of this group of composers was based on desktop computers connected to a local network. Each member of the group was responsible for some musical feature during the performance and the interaction between them would take place through the local network, and later, through phone lines. Many other network pieces have been attempted in the context of laptop orchestras, including the Princeton Laptop Orchestra (PLOrk) [12], the Stanford laptop orchestra (slork) [13] and Linux Laptop Orchestra (L2Ork) [14]. All of these orchestras focus on local network solutions for communication and have an infrastructure for small ensembles. Additionally, we have some works where the performers use local network to share data and communicate within the ensemble [7], [15]–[18].

Following these ideas, one can cite the Republic [9] quark package that is used to create synchronized network performances. In this case, the live coders would have to start a server and have all users connected in order to start sharing code and interacting. Another work that have similar functionality is the extramuros [10], a system for network interaction through sharing buffers of any kind of language. The last solution needs to be configured depending on the language and at the time of this paper it does not present any easy way to stop synthesis on SuperCollider. Both solutions requires a server on one computer to receive connections from clients, and additionally it would be necessary to open network ports or change firewall settings before starting any interaction.

The collaboration in live coding is also presented on the Gibber[11] library for WebAudio. The users can synthesize code on the browser and talk through a chat room in other to have a collaborative online session. Although this solution permits users to share code and synthesize online, there is no way to share the same code environment at the same time, and the users will probably need to share the code through the web chat if they want to try a live session. In the session we will then present SuperCopair, a tool created by the authors that can fulfill all the blank spaces for collaborative live coding and also inspire the cooperative live coding practice.

IV. SUPERCOPAIR

SuperCopair is an application created as a package for the Atom.io[12] IDE. This IDE has numerous packages for many programming languages and presents some solutions for coding, debugging, and managing projects. Atom packages are programmed in CoffeeScript[13], which is a programming language that can be converted to Javascript and can also integrate its libraries. The developers can install Atom packages to enable various functionalities in the IDE such as: communicate through chats, use auto-complete in certain programming language syntax, interact with desktop and web applications, integrate with the terminal command line, and have many options based on other packages. In the same way, the users can just search for SuperCopair and install without many steps.

SuperCopair is based on two Atom packages: atom-supercollider and atom-pair. The first package turns Atom.io into an alternative IDE for SuperCollider programming language and permits users to openly communicate locally with SuperCollider audio server through OSC in the same way we can do on SC-IDE, the default IDE. Additionally, the users can take advantage of packages from Atom and quarks together in the same interface. The latter package is used for pair programming through the Internet. The atom-pair package is based on Pusher cloud service and its default configuration is based on the community free plan, but a user can modify the settings and use his/her own keys. We decided to merge both packages to add new features for collaborative live coding, and finally had dubbed it the SuperCopair package. The main idea is that all participants evolve into a collaborative practice and performance.

The IDEs for SuperCollider have shortcuts to evaluate a line, a block, and to stop all sound process that is running. In addition to these options, the SuperCopair package includes methods and shortcuts that can broadcast the events cited and execute them on all users connected at the same pairing session. Through the shortcuts, one can decide to evaluate selected code either only in the local machine or in all computers on the same session. One can also turn on and off a broadcast alert option in the settings in order to be asked or not before evaluating every broadcast event sent by another user in the same session. These options allow each individual to have control over the code that can be evaluated in the local machine.

The broadcast events are diffused to all connected users through the cloud service and the package evaluates on arrival each event message received. The message includes the code to be evaluated and the user identification. A representation of a session using SuperCopair package is shown at Figure 1.

Before inviting others, users need to start a new session following an instructional step-by-step setup presented on the package page. Once a session starts, the session ID string needs to be shared between collaborators that want to join the same session. The shared session ID is based on the channel created at the cloud service and it contains the user's keys. Every user who joins a session will see the most recent version of the shared code. The users can identify each other by different color markers on the left side of the shared file representing the line or lines in edition. A pop up provides information about users joining or leaving the session. Furthermore, a message including user's identification and the code evaluated appear at SuperCollider post window right after each broadcast event is evaluated. In case the broadcast alert option is on, a dialog will appear whenever the user receives an event message from another participant and this dialog asks if the user would accept or reject the code evaluation. The alert dialog will have the sender's id and the code sent via broadcast.

To summarize, this tool permits both participants to share a distributed audio environment and listen to the same sound when synthesized in a broadcast manner. SuperCollider live

[9]Republic quark: https://github.com/supercollider-quarks/Republic
[10]extramuros: https://github.com/d0kt0r0/extramuros
[11]Gibber: http://gibber.mat.ucsb.edu/
[12]Atom.io: http://atom.io
[13]CoffeeScript: http://coffeescript.org

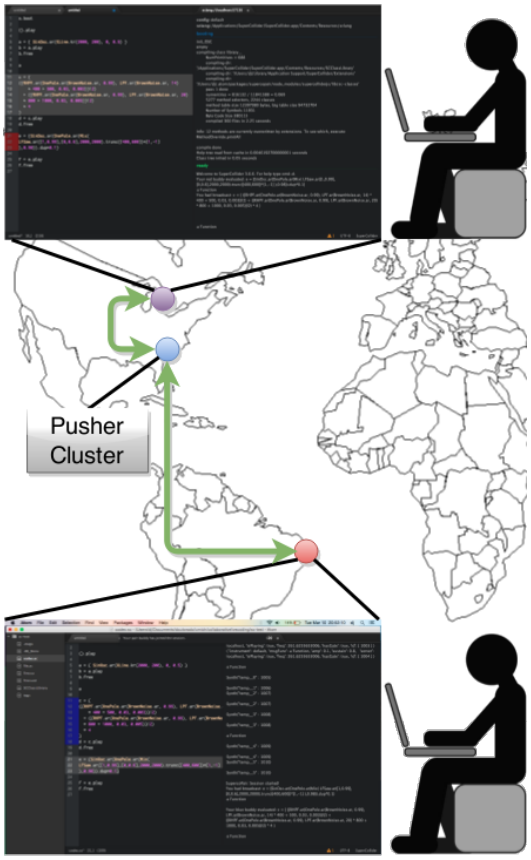


Fig. 1. Session with two users from different countries.

coding sessions through SuperCopair will take advantage of the easy setup and the minimal requirement of an Internet connection, instead of the network and server configuration required by the other solutions cited. Although the main idea behind this project is to provide a tool for collaborative live coding, new usability is proposed in this paper as a result of reflection after some experiments. In the next section we are going to discuss the possibilities of learning program languages related to music and the new paradigms focusing on cooperation and musical interaction.

V. Cooperative live coding

The cooperation and collaboration terms are similar regarding the idea of working together, but they differ in terms of benefits offered and tasks equilibrium. Inside a collaborative environment, we have all participants focusing on the same goal and working together to achieve same objectives. One can suggest that the players of the same team on any sport game are working collaboratively in order to win the game. On the other hand, a cooperative environment provides different benefits for the participants and they may have dissimilar objectives. Following the last example, two teams might cooperate during a match following the rules of the game in order to have a fair play, but they will not share the same goal (because one has to win) and each team probably benefits from the other's (failed) activities.

The cooperation can also be discussed in other ambit like: biology, when some species interacts in mutualism; business, when one company offer a service to the other, even if they share the same market; and personal life, when a relationship end up in divorce due to the unsuccessful collaboration between the participants or to the absence of shared goals. These examples illustrate the many characteristics of the cooperation and how it can be applied in different areas. In this section we will discuss the aspects of cooperative live coding in computer music education.

Networked live coding environments have been focused in collaboration as we presented on Section III. All participants are working together to execute a musical piece and share the same final result as a common goal. In most of the cases, sound is the only artifact that is shared, but the performers can also share codes. Some tools also enable chatting, audio, and video interaction between the participants, whether they are sharing the same physical space or they are in different locations. Although the experiments with SuperCopair have been conducted as distributed collaborative live coding sessions, the cooperative environment has emerged during some moments.

During the sessions using SuperCopair, we had participants from different levels of experience on live coding and also some new users that decided to learn how to code during the session. The collaboration on the final sound was similar to other live coding sessions, but the experience added another way of interaction in live coding: the cooperation.

The cooperation aspect emerged from some events that came into view from the live coding sessions. Experienced users are faster and have written lots of code from the scratch without any problem, while the apprentices start from basic structures or from portions of codes available on the file. As all users were trying to synthesize the codes on all computers, it became easy to perceive if something was going wrong because everybody was sharing the code, evaluation, synthesis, and errors. The errors coming from novice programmers were often fixed by the experienced users, and they also discussed the solutions between themselves using comments on the file. Some tricks from the live coding practice were introduced by advanced users and the initial learners rapidly became instructors for new users and these instructors were following the same cooperative practices of the experienced users. These events inspired the instructional model that is proposed and described in the next section.

VI. The instructional model

The discussion about the instructional model starts with an example of a cooperative live coding session that is presented on Figure 2. In this session we have two users working on the same file through the SuperCopair package on Atom.io IDE. Both users can see the line that is being edited by the other user following the color mark at the line number column. The comments presented on the file explain the following codes.

In this live coding session the users have the same benefits of any online environment for PP or DPP. However, this session presents more benefits and advantages due to the cloud service and IDE integrated through SuperCopair.

While some solutions presented at Subsection II-B have cloud services on their background, the cloud service used in our solution offer an adaptable and easy-to-setup environment for multiple users. Users can pay for dedicated infrastructures in order to avoid the free service limitations, and it is also possible to select the preferable cluster to connect. The network administrators from the service will take care of data distribution system which has a high level of abstraction from the user point of view. A key from the service is all the client need in order to use the service requested and it encourages the adoption of this service by novices and advanced users.

The IDE adopted includes many features that will hardly be available at the solutions discussed at Session II. There is a community working on packages for this IDE, making lots of options available, and supporting the development of packages like SuperCopair. Atom.io IDE offers common features like code completion, many shortcuts, and code highlight, but it also presents almost three thousand packages available with other features at the time of this paper.

A. Educational aspects

Although the cloud service and IDE bring many advantages for this live coding session, the combination of their features imply a new educational concept for computer music and live coding that we introduce below. We also have a discussion regarding educational values from the instructional model proposed herein. All the instructions and directions can be applied to local or local network structures as the instructor and learner can share or not the same classroom during the process. We can also consider a mixed environment where a group of participants share the same place while other participants are remotely connected to the same session.

1) Use comments for discussion: In this instructional model, the learner and the instructor can write comments with questions or instructions on the file. Comments are useful in this case because they will not be interpreted or cause errors even if the whole file is evaluated. The communication through comments is important if the instructor and the learner share the file from different locations, but the comments may be avoided whenever all users are in the same room.

2) Assist and fix codes: The cooperative live coding is also an advantage for collaborative live coding performances. An experienced user of the language can audit the performance and help to fix codes while other are performing. This special user does not need to participate in the performance as musician, and can also act just as a conductor writing comments in order to guide the live coders during the performance. The users of forums and mailing lists for SuperCollider questions can schedule meetings to code together. The sessions are not restricted for two users and the final code can be posted again in the same place where the discussion has started.

3) Interact during classes: Programming classes can include more interaction if all students can access the same file as the instructor. The code can be synthesized on all computers at the same time, and the students can hear the sound at their own computers without great speakers on the classroom connected to the computer of the instructor. Some students can make use of headphones when preferable, and the number of students connected is not a problem due to the advantages of the cloud computing behind the system. Both students and instructors can participate on the class in this model even if they are at home or traveling to a conference. Additionally, the students can interact directly and instantly on the code learned, as they only need to add code or comments from their own computers during the class. This model can also provide benefits to the instructor, which can ask questions for the students and see the code being written lively on the file shared.

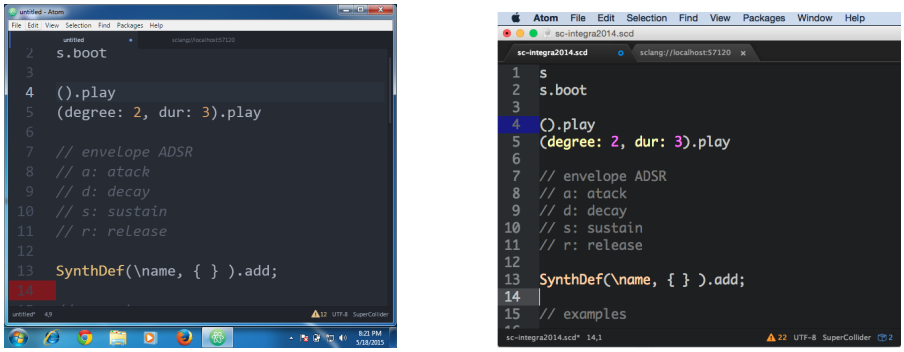
4) Offer online remote tutoring: There are students that prefer personal instructors as a supplement to the learning process, and some instructors prefer to work from home or want to have students from different locations. In this case, both can have benefits from this model. The instructors will not need to move between the students houses and can schedule more classes without intervals between them taking advantage of the interaction through the Internet, and the students can have instructors from different countries. The best students can also offer assistance to others after classes even without an specific place to meet together. On the other hand, experienced live coders can share their knowledge at workshops without physical participation, while the audience can join the same session, share the code, and ask questions on the file The audience can also be distributed, and the proposal of workshops will make the most of the new technologies exploring the functionality of the tool presented.

Although it is not an exhaustive list of directions, we intend to consider this discussion as a starting point for this instructional model for computer music and live coding through a cloud service solution. This instructional model have the cooperation in favor of the instructor or the learner depending on the situation. The participants may not have the same benefits or goals during a cooperative live coding, but they have to evolve in a helping behavior with or without reward in order to agree in a cooperative live coding environment. An open cooperation movement can also arise from this practice and confront some limits of learning.

VII. Conclusion

Many instructional models have been proposed since the first advances on communication technologies. We have instructors using letters, magazines, calling, videos, and including occasional meetings to the teaching process, or at least using one of these channels for receiving feedback from learners. What is important is the link between the learning material and the learning process to ensure the nature of the distance learning models [19].

In this paper we described a new model of distance education that can make use of the distributed live coding and introduce the cooperative idea. In this model we propose that instructors and learners can share codes and audio synthesis from distant places and also in the same room through the advantages of SuperCopair package. The cloud computing behind the solution allows participants to join a session and interact through the file where the code is being written almost



(a) Windows user (b) Mac OS user

Fig. 2. Cooperative session with two users. The users share the same file and they can see where the other pair is working following the colors markes visible at the line number column. They can also write comments on the file in order to communicate.

instantly, and without problem with the number of participants. As for example, the free plan used at the cloud service behind SuperCopair accepts up to 20 users, but one can have 10 thousand participants when assigned for a paid plan.

The cooperation of the participants is assumed during the live coding session, although new features can be added in the future in order to add more control from the instructor's point of view. We suggested many opportunities for the practice of cooperative live coding following our experiences, and one can conclude that the results can be gradually presented. During our sessions, the learners became instructors of basic topics in a short time. The interaction through the comments was valuable like a chat with many rooms, as we had many discussions in many portions of the file happening at the same time.

The tool is proposed to SuperCollider, but it can be extended to languages like Csound, Chuck, and Processing. Interpreted programming languages are more suitable for this environment due to the fast output without compilation. However, the idea of cooperative live coding can be used with any programming language, considering that the compilation on the learner side needs to be done depending on the operating system.

The authors wish that the cooperative live coding instigate many instructional models, and that more solutions for interaction between distant learners and instructors emerge from the advances on cloud computing. As the Internet is the only requirement for this practice using , we need to focus the attention on bandwidth, timezone, and system requirements in order to have more cooperative live coding sessions.

REFERENCES

[1] C. V. A. P. da Silva, C. B. L. Neto, and M. R. Petrucci, "Hipermídias educativas: a aplicabilidade de objetos de aprendizagem em planos de aula nos espaços virtuais," in *IV Encontro Nacional de Hipertexto e Tecnologias Educacionais*. Universidade de Sorocaba, 2011.

[2] M. Silva *et al.*, *Sala de aula interativa*. Quartet, 2000.

[3] Â. Á. C. Dias and H. Chaves Filho, "A gênese sócio-histórica da idéia de interação e interatividade," *Tecnologias na educação e formação de professores*, 2003.

[4] S. M. Hord, "Working together: Cooperation or collaboration?." 1981.

[5] B. J. da Silva Estácio and R. Prikladnicki, "Distributed pair programming: A systematic literature review," *Information and Software Technology*, vol. 63, no. 0, pp. 1 – 10, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584915000476

[6] P. Ford, "Processing processing," in *The Best Software Writing I*. Springer, 2005, pp. 79–93.

[7] N. COLLINS, A. McLEAN, J. ROHRHUBER, and A. WARD, "Live coding in laptop performance," *Organised Sound*, vol. 8, pp. 321–330, 12 2003. [Online]. Available: http://journals.cambridge.org/article_S135577180300030X

[8] N. Collins, "Live coding of consequence," *Leonardo*, vol. 44, no. 3, pp. 207–211, 2011.

[9] T. Magnusson, "Algorithms as scores: Coding live music," *Leonardo Music Journal*, vol. 21, pp. 19–23, 2011.

[10] C. Nilson, "Live coding practice," in *Proceedings of the 7th International Conference on New Interfaces for Musical Expression*, ser. NIME '07. New York, NY, USA: ACM, 2007, pp. 112–117. [Online]. Available: http://doi.acm.org/10.1145/1279740.1279760

[11] G. Weinberg, "The aesthetics, history, and future challenges of interconnected music networks," in *International Computer Music Conference*, 2002, pp. 349–356.

[12] D. Trueman, "Why a laptop orchestra?" *Organised Sound*, vol. 12, no. 02, pp. 171–179, 2007.

[13] G. Wang, N. Bryan, J. Oh, and R. Hamilton, *Stanford laptop orchestra (slork)*, 2009.

[14] I. I. Bukvic, T. Martin, E. Standley, and M. Matthews, "Introducing l2ork: Linux laptop orchestra," in *New Interfaces for Musical Expression*, 2010, pp. 170–173.

[15] J. Rohrhuber, A. de Campo, R. Wieser, J.-K. van Kampen, E. Ho, and H. Hölzl, "Purloined letters and distributed persons," in *Music in the Global Village Conference (Budapest)*, 2007.

[16] A. R. Brown and A. C. Sorensen, "aa-cell in practice: An approach to musical live coding," in *Proceedings of the International Computer Music Conference*. International Computer Music Association, 2007, pp. 292–299.

[17] S. Wilson, N. Lorway, R. Coull, K. Vasilakos, and T. Moyers, "Free as in beer: Some explorations into structured improvisation using networked live-coding systems," *Computer Music Journal*, vol. 38, no. 1, pp. 54–64, 2014.

[18] D. Ogborn, "Live coding in a scalable, participatory laptop orchestra," *Computer Music Journal*, vol. 38, no. 1, pp. 17–30, 2014.

[19] D. J. Keegan, "On defining distance education," *Distance Education*, vol. 1, no. 1, pp. 13–36, 1980. [Online]. Available: http://dx.doi.org/10.1080/0158791800010102

F.10 WAC 2016 - Crowd in C[loud]: Audience Participation Music with Online Dating Metaphor using Cloud Service

Paper details

Title: *Crowd in C[loud]: Audience Participation Music with Online Dating Metaphor using Cloud Service*

Authors: Sang Won Lee, Antonio Deusany de Carvalho Junior, Georg Essl

Conference details

Title: 2nd Web Audio Conference (WAC)

Venue: Georgia Tech, Atlanta, GA, USA

Dates: April 4 to 6, 2016

Crowd in C[loud] : Audience Participation Music with Online Dating Metaphor using Cloud Service

Sang Won Lee
Computer Science and Engineering
University of Michigan
2260 Hayward Ave
Ann Arbor, MI 48109-2121
snaglee@umich.edu

Antonio Deusany de Carvalho Junior
Universidade de São Paulo
Rua do Matão, 1010, São Paulo, SP, Brazil
dj@ime.usp.br

Georg Essl
Electrical Engineering & Computer Science and Music
University of Michigan
2260 Hayward Ave
Ann Arbor, MI 48109-2121
gessl@umich.edu

ABSTRACT

In this paper, we introduce *Crowd in C[loud]*, a networked music piece designed for audience participation at a music concert. We developed a networked musical instrument for the web browser where a casual smartphone user can play music as well as interact with other audience members. A participant composes a short tune with five notes and serving as a personal profile picture of each individual throughout the piece. The notion of musical profiles is used to form a social network that mimics an online-dating website. People browse the profiles of others, choose someone they like, and initiate interaction online and offline. We utilize a cloud service that helps build, without a server-side programming, a large-scale networked music ensemble on the web. This paper introduces the design choices for this distributed musical instrument. It describes details on how the crowd is orchestrated through the cloud service. We discuss how it facilitates mingling with one another. Finally we show how live coding is incorporated while maintaining the coherence of the piece. From rehearsal to actual performance, the crowd takes part in the process of producing the piece.

1. INTRODUCTION

Web Audio significantly lowers the level of complexity for creating networked interactive music application and cloud technologies allow for creating scaleable audience participation in this context. In this paper, we introduce a web-based audience participation implementation as used in the music piece, *Crowd in C[loud]*. A distributed musical instrument is implemented entirely for a web browser, enabling an audience to easily participate in music making with their smartphones. This web-based instrument is designed to encourage the audience to play music together and to interact with other audience members. Each participant composes a short musical tune that serves as a musical profile of that particular participant. Once a profile is submitted, they can browse other people's profiles and play a pattern they like forming pairs. These musical profiles serve a metaphor for online dating websites.

To realize the network capability of the musical application, we made use of a cloud service to exchange data. It allowed us to create a networked ensemble without the hassle of configuring and developing a server program. A computer mediator and performer can actively progress the music by orchestrating the crowd through live coding on the console of the web browser.

In this paper, we describe the inspiration for musical aesthetic of the piece, justify the design choices that we make, and introduce the technical details of the instrument. Lastly, we share our experience of the rehearsal process and describe the performance of the piece.

2. BACKGROUND

Mobile smart phones are an attractive platform to enable audience participation in musical performance, as today it is sensible to assume that many audience members do have their own mobile device at hand. Realizing contemporary audience participation can has taken two routes: Developing native applications or using web technologies.

In recent years, developing (or repurposing) a native application for smartphones has been a common approach [17, 21, 25]. This is attractive because musicians and developers can design an instrument choosing from a full range of interactivity and can utilize the full computational power of a mobile phone. However, such an application often limits participation to a set of people who use a certain operating system (e.g. iOS). In addition, there are the challenges of downloading a native app and setting up the network configuration inside the app.

On the other hand, the web browser is a popular choice for audience participation because it alleviates the aforementioned problems; no additional installation is required, the web browser runs on multiple platforms and it is easy to distribute or update an application. It has been used in numerous previous works even before the time of Web Audio. Freeman's Graph Theory [11] and Piano Etudes [12] constitute such examples. In these cases, participants directly and indirectly took part in the process of composing music piece via web browsers prior to a live performance. Instead of customized websites, a pre-existing social network system (e.g., Twitter) that runs on any web browsers can be re-purposed for real-time participation [8].

massMobile, a general framework for audience participation, facilitates rapid development of various participatory

Web Audio Conference WAC-2016, April 4–6, 2016, Atlanta, USA.

applications and enables plug-and-play setup on mobile web platform [29]. Another framework, SWARMED, implements a captive portal so that any audience member can, with minimal configuration, connect to the web-based musical interface [14]. Early usage of web browsers in audience participation without web audio was limited regarding native sound synthesis on mobile devices. In that sense, the audience acts as a composer, influencing the piece on stage, rather than a performer generating sound from a mobile phone [21].

In the tradition of network music, users have deployed web browsers in the collaborative music making endeavors [6, 7]. The Web Audio API [26] accelerated emerging trends of web browser-based music applications where sound is synthesized and generated directly from web pages.

Recent efforts push the performance of web-based audio applications towards the level of native audio applications [22, 5]. The majority of music performances presented in the first Web Audio Conference involves audience participation (or audience involvement) [1, 18, 24, 27, 28]. This reflects the web audio community's strong focus of collaborative music making with the audience. This mobile approach differs from previous approaches where the audience influences music indirectly and sound comes from a stage. *Crowd in C[loud]* draws upon the ideas of existing audience participation works and introduces a new venue for networked web-based music app using a cloud service.

Previous works in audience participation have often relied on a local network with a server developed, configured, and managed by the developers [17, 29, 14, 21]. The use of a cloud-computing infrastructure in the context of computer music requires no physical server on the spot and the cloud can be used in various settings — remote-networked performances, online collaboration systems, audience participation, and locally networked ensembles.

A series of tools to support cloud computing in the context of collaborative composition. Computer Music Cloud is a system for music composition in the cloud with a data exchange protocol [2]. A textual representation of music notation [3] and a computing architecture designed for online music composition system [4] further add to this tool set.

Hindle's "CloudOrch" [15] presented a system that implemented a virtual soundboard on the cloud and, without a physical soundboard or mixer, interconnected multiple audio inputs and outputs. Using the cloud soundboard frees a musician from carrying about high performance machines for computationally heavy music performances. The author conducted a follow-up study in the deployment of such scaled music applications and resource allocation for many computers in cloud [16].

As opposed to using cloud servers, a commodity cloud service provides a convenient option to build a real-time network among computers. It makes the network configuration abstract to users and can be used to distribute data via cloud data centers located globally. An evaluation the efficiency of a cloud service in computer music applications sending control signals measured latency (83ms) between devices located in North America and South America through the Pusher[1] cloud service [9]. Using the same cloud service, SuperCopair provides real-time shared document support for multi-performer live coding in Supercollider [10].

[1]Pusher cloud service: http://www.pusher.com/

3. CROWD IN C[LOUD]

As the name of the piece suggests, the crowd (audience) plays the musical instrument in C Major. This is directly inspired from the piece *In C*, by Terry Riley [23]. In this piece, musicians (with various instruments) were guided to play pre-composed melodic fragments in sequence for a random amount of time. As it is up to each musician to decide how many times to play one fragment, the collective outcome of the ensemble creates a heterophonic texture of chance. Similarly, in *Crowd in C[loud]*, each audience member plays a series of short snippets composed by herself and by other audience members. The interface provided will first guide a participant to compose a short `"tune"` that has five musical notes in C major. Once the participant finishes the composition, he or she can browse, and play, what other audience members have composed. It is thus quite similar to Terry Riley's *In C*, in that one determines for oneself how long to play a tune. The difference is that there is no pre-composed fragments but each audience member will contribute to the piece by submitting a short melody. In this way, participants will have their own tunes and a chord scale become the common ground upon which the entire audience plays. In addition, there is a separate musician performing the piece on stage at the same time with the audience members in *Crowd in C[loud]*. The role of the musician is a meta-performer who can control the chord scale in which the audience members are playing. For example, the meta performer can, on the fly, change the instrument tuned in C major scale to a different chord scale (e.g., C Minor, Pentatonic Scale). This performer cannot generate sound at all on his/her end but only controls the harmonic flow of the piece as generated by the crowd. The interplay between the musician and audience members ensures that each audience member will play individual patterns while a musician can progress the piece by changing chord scales. This performer-audience pairing model comes from a previous work of *echobo* [21] where audience members played a simplified key instrument on smartphones with the chord progression determined by a performer on stage and synchronized over a mobile network.

3.1 Loop-based Instrument

The web-based musical instrument we developed for the piece contains a simple interface that can loop a five-note melody in a specified scale (C major scale in the beginning). There are five circular notes (or "`note dots`") that are connected by lines. And there is a circular play head (or "`the play dot`") in yellow which travels over five red (or green) note dots, triggering a tone whenever it reaches a note dot. The play dot moves at a constant speed so that a melodic pattern (or "`Tune`") will be looped consistently. This ensures that the instrument will generate sound without any user involvement as long as the user turns up the volume and stays on the page. This way, the musician need not worry about being too sparse or silent due to low participation. The expressive range of the instrument depends on where a player places note dots on a screen. First, the vertical position of note dots determines the pitch of the notes. The interface visualizes pitch difference with alternating white and gray divisions in the background. Secondly, the horizontal position of a note dot determines the timbre of the tone; the note dots placed on the leftmost side of the screen will generate pure sine tones while the note dots on the other

end will play a tone that combines various oscillators (sine, sawtooth, square, and triangle waves with different detune parameters).

Sound synthesis of the instrument is entirely realized using Web Audio API oscillators. The instrument implements a JavaScript object for each tone (or "`voice`"). Each time the play dot reaches a note dot, the program creates a voice instance that contains a set of oscillators. The voice instance includes a JavaScript object that contains a Gain node and implements an ADSR envelope. The interval between the two consecutive notes is determined by the length of the line in between and the duration of each note is proportional to the interval. A tune can be archived with the position data of five note dots in order and the archived data can be later shared with other audience members to reproduce the tune in other devices.

Note that the duration of one's tune can be arbitrarily long and is not exactly the same as the tune of other audience member. We embrace that asynchronicity among ensemble members and leave the temporal expressivity of a tune up to each player. It is similar to the original version of Terry Riley's *In C* where there was no pulse. We find that having the synchronized global pulse and quantized beats gives the audience a different experience and achieves a different style of music, the exploration of which we leave to future work.

We wanted to design the instrument to be extremely accessible so the audience to pick it up in a few seconds but still be good enough for a participant to be musically expressive. However, simplicity can induce limited flexibility and expressive range and hence hamper long-term engagement. Indeed, the mixed use of note dot locations for multiple parameters (timbre, pitch, and time) constrains the expressive space of the instrument. For example, one cannot play two consecutive notes of the same timbre and same pitch with a long interval in our interface. While we could have made an effort to build a musical instrument that achieves *low entry and no ceiling* [30], we take a different approach to encourage the participation. We find it acceptable to develop a constrained musical instrument [13] in the hope that participants will discover a diversity of playing style via social interaction with other audience members.

3.2 Online Dating Metaphor

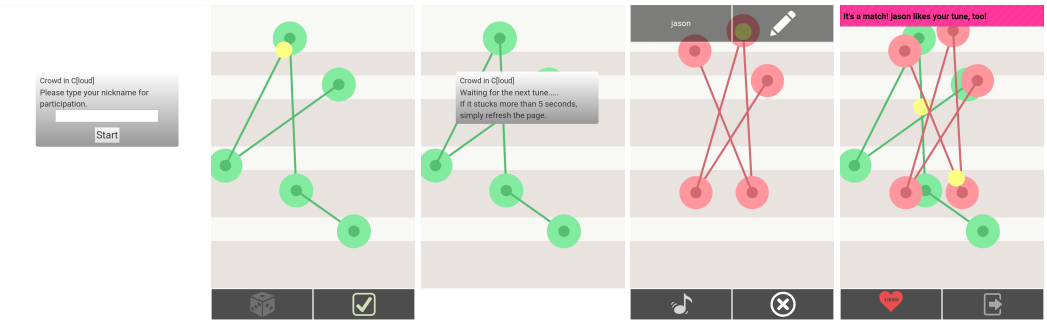
As discussed earlier, the musical instrument provided to audience members has clear limitations; it can only loop five notes with different pitch choices and timbre variations. In turn, once the user finishes the composition of a short snippet, the user is able to browse other people's composition. This grows out of the idea of an online-dating website (such as Tinder) where a user creates a personal profile and then browses other member profiles that include pictures and written descriptions about themselves. Similarly, the networked instrument creates a temporary social network that will last until the end of the performance where a short tune is used as a musical profile. Lastly, the collection of each tune composed by individuals serves as musical phrases such as found in Riley's *In C*. The difference here is that the number of musical phrases is the same as the number of participants and each participant can change the short composition on the fly.

Allowing participants to browse other tunes is expected to motivate people to play the instrument in various ways. First, it motivates a user to compose a tune to express oneself, to attract more people, and to find a (musical) match among the participants. This resembles a self-presentation strategy in online dating sites where participants post photographs and a written description that represents themselves well. Secondly, browsing the tunes inspire participants to discover new styles with which to play the instrument. For example, suppose one created a tune with ascending tones in C major and then later discover a tune that uses note dots to visually draw a certain object. Later, one may find another tune that have five note dots in one place close enough so that it creates a very dense rhythmic pattern. Lastly, we bring the joy of playing together. In `MINGLE` mode, discussed below, one can play his or her own tune with another tune. When two tunes are looped on the same screen, a user is allowed to modify his or her own tune to musically match that of the other. It is a metaphor for the situation where two people on an online-dating website start a conversation, meet off-line, and explore the possibility of being a match. We are planning to analyze the interaction of the participants to investigate whether this socially connected ensemble actually inspired each other.

While there are many different levels of interaction in online-dating websites, we borrowed the simplest model from a popular online dating application, Tinder. On Tinder, one can browse profiles, press the like button, and start to chat when it's a match. The musical instrument can be in one of the five different states: `NAME`, `EDIT`, `WAIT`, `CHECK`, and `MINGLE`. Each state is used to design different interfaces and determine what other states a user can reach out from and go to. The five states are described below.

- `NAME`: When an audience member first visits the link provided http://bit.ly/crowdinc, the member is prompted to type a unique screen name that will be used throughout the performance (Figure 1a). Once the participant submits a valid screen name, the web page will be redirected to the `EDIT` state.
- `EDIT`: A participant composes a tune in this state by dragging and dropping the note dots. The play dots will continue while editing so one can hear the current tune (Figure 1b).
- `WAIT`: This is a transient state where the instrument is waiting for a message from the cloud service after a request for data. The incoming message contains data for another member's tune (Figure 1c).
- `CHECK`: This is a state where a participant can browse the tunes of the other members (Figure 1d).
- `MINGLE`: This is the state where a participant can play two tunes at the same time (Figure 1e). The note dots in green are the tune composed by the user and the note dots in red are the tune composed by another audience member. In this mode, one can freely move green note dots to make two tunes sound differently and explore a new musical pattern with the combination. The red dots cannot be modified.

The interface is designed to notify social interaction by broadcasting messages. For example, when a participant named John "likes" Jane's musical profile by pressing the heart-shaped button in `MINGLE` mode, Jane will receive a



(a) NAME state (b) EDIT state (c) WAIT state (d) CHECK state (e) MINGLE state

Figure 1: Screenshots of Audience Interface in Five States.

message saying "John likes your tune!" Later, Jane browses more tunes and "likes" John's pattern. John will then receive in the top banner saying "It's a match! Jane liked you back!" (Figure 1e).

On the other hand, the performer's interface (Figure 2), which is also a web page, is used to display the list of screen names that are currently participating in the performance. The performer program calculates the number of likes received and the number of participants playing the pattern at the moment under each individual screen name. This interface works like a score-board when projected on a screen at a concert hall. On the top right corner, it shows the screen name of the participant whose tune is most liked and the screen name of the participant whose tune is played most at the moment (named "most crowded"). This projection helps audience members realize that the nature of the participation is social and it also helps a non-participating audience engage with the piece by looking at how their friends and families are doing.

4. CROWD MUSICKING IN CLOUD

The performer interface and the audience interface are available at the following sites. Performer : http://bit.ly/performerinc, Audience : http://bit.ly/crowdinc. To try the demo, press the "`Go Live`" button in the performer interface and use multiple devices to play in the audience interface. Currently there is only support for a single performer using the performer interface.

4.1 Network Structure - Cloud Service

We utilized a cloud service to exchange data among audience members and to orchestrate a chord scale of the crowd. The performer interface and the audience interface are two static web pages hosted on a university web server. Once both web pages are downloaded to a device, there is no dynamic interaction between the device and the web server. The performer interface runs on a laptop and the audience interface typically runs on a participant's smartphone. The performer interface maintains the relevant data (in local JavaScript data structure) regarding all the participants' data (tunes) and all the information needed to display on the scoreboard. Although the performer interface is a web page running on a local machine, it acts as a server in the traditional sense. The only difference is that the server (a performer's laptop) and the clients (audience's smartphones) communicate via a cloud service with minimal network configuration, which is already hard-coded inside the JavaScript file.

After comparing many cloud services, we chose the PubNub cloud service.[2] PubNub provides better bandwidthand reliability.While we used a free plan from PubNub for the development and rehearsal, we purchased the cheapest paid plan to obtain a dedicated key that will allow more than 100 participants in on the session for the actual performance.

PubNub follows a pub-sub paradigm for data communication in JavaScript. Any number of JavaScript web application using the same application key can publish (or send) messages to certain channels or subscribe (or listen) to one or more channels. There are three types of channels used in the application — `performer`, `audience`, and `<uuid>`. `performer` is a channel that only the performer program listens to and it is used when audience programs make a request to retrieve certain data such as a tune object. `<uuid>` stands for universal unique id and is given to each participant as soon as the page is load on the device. It is used to transfer data from the performer to each individual, which is the response to the request mentioned previously. Lastly, all clients subscribe to the `audience` channel and it is used to broadcast a message or to change the chord scale of participants instruments.

4.2 Orchestrating the Crowd via Live Coding

Changing the chord scale in an audience application is pre-written as a JavaScript function and the performer can send a signal to the `audience` channel to call up the function to make changes in the entire crowd. While there are many ways to signal the function call in audience members' devices (buttons, knobs, sliders, keys), we chose to live code on the JavaScript console of the web browser. A performer can type the following line on the console to change the chord scale of the whole crowd.

```
publishMessage("audience",type:"scale",baseNote:60,scale:[0,3,7,12]);
```

`publishMessage` function is written to broadcast a message with a JavaScript object to a specified channel (`audience` in this case). `type` indicates that the message is to change the scale; the parsing function in the audience

[2]PubNub Cloud Service: http://www.pubnub.com

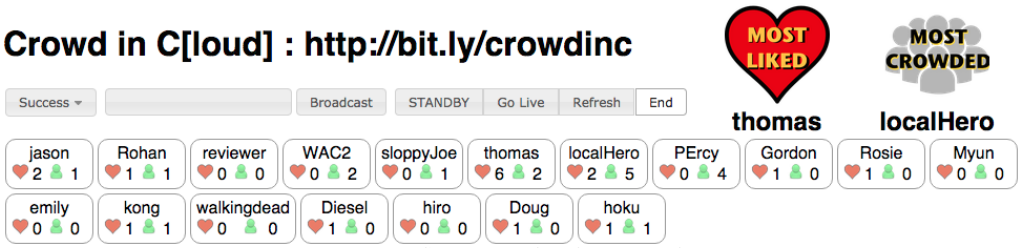


Figure 2: a Screen-shot of Performer Interface

application expects `baseNote` and `scale` within the object. And then the function call in the audience JavaScript program will change the scale to C Minor $(C, E\flat, G, C)$ starting from middle C, of which the MIDI note number is 60. Whenever a performer sends this kind of scale-control message, they are informed with a message on the top banner that, in a few seconds, disappears automatically.

Live coding JavaScript code on the console expands the flexibility of what a performer can do on the crowd's machines. For example, a performer can send any kind of text string that can be evaluated in a JavaScript application on the audience side using a `script`-typed message. See the following three examples:

```
publishMessage("audience",type:"script",script:"soundEnabled=false;");
publishMessage("audience",type:"script",script:"refresh();");
publishMessage("audience",type:"script",script:"alert('hello');";
```

The first example will set a variable `soundEnabled` to false, which is a boolean variable in the audience program state to determine whether to switch on/off the sound synthesis (all device will be muted!). The second example will run the function `refresh();`, which is readily available in the audience program to refresh the page (so everyone is forced to start over!) The third example, maliciously enough, will show an alert box on all smartphones and the web audio synthesis will be halted until the user clicks the okay button.

In orchestrating the crowd, a musician may want to change only the subset of the crowd to produce diverse sound, for example, half the audience playing in C Chord the other half playing the F note only. We included the `probability` property used with `scale`- and `script`-typed messages to achieve a partial code run. If the performer includes `probability : <a float number>` to the JavaScript object in messages as above, it will run the parsed action with the probability of the given number. For example, if `probability : 0.5` was attached at the end of an object, the code will run with a 50% chance, so that the performer can make only (roughly) half of the crowd take the change. The model of one live coder controlling crowd-scale computer networks was proposed in [20] and we believe this is the first realization of the idea. While the potential of live coding has not yet been explored other than changing the scale, we believe there are novel musical styles and aesthetics that we can achieve with this live coding of large-scale machines. We plan to use this feature to live code the web-based instrument [19] to achieve diverse styles of music by changing more than just the chord scale (timbre, interface, mapping), while leveraging human computation of individual users for musical expression within this dynamic scenario.

5. CROWD IN C[LOUD] IN ACTION

We premiered *Crowd in C[loud]* at the Winter Final Class Concert [3] in Stamps Auditorium at the University of Michigan North Campus. The audience consisted predominantly of students and local citizens, who had little background in computer music but were casual smartphone users.

The program note included a shortened link (bit.ly and QR code) and a set of step-by-step instructions that described how to participate so the audience could access the web page and try the interface before the concert began. As long as the participants had devices that could run a web-audio-enabled web browser with any connectivity (mobile or WiFi), they would be able to participate in the piece. There were no additional steps needed such as joining a designated WiFi network, downloading a native app from app stores or typing an IP address, which for casual users may be challenging.

For the performance, the first author composed a short piece and played the role of performer. The performance was started by the performer giving a sign to the audience and pressing the "Go-Live" button at the top of the interface. For the first few minutes, the performer did not intervene to change the global scale. Rather, he communicated with the audience broadcasting chat messages to explain the instrument (timbre, pitch mapping), to encourage participation, and to introduce what the projection screen showed using the message broadcasting. The performer started to change the scale occasionally for the latter part of the performance. The performer interface included frequently used code in a textarea so that the performer could quickly copy and paste in a contingency of possible errors in typing code or scale. At a certain point, the performer changed the global scale to one note so that all device would generate a one-pitched sound. This was to facilitate the playing of a simple melody by quickly running the series of code that had different base note numbers. Later, the crowd was divided into two groups using the `probability` option and one group played a sequence of unified notes while the other half played a background chord.

The performance was well received by the audience and we got positive feedback an various levels. Video footage showed participants, ranging from young kids to the elderly, actively engaged with the instrument and occasionally watching the projection screen.

[3]The video footage of the first performance are available in two following links.
https://youtu.be/wCXhGotDtFs?t=248 : audience seats
https://youtu.be/8RWgXoM2BCA?t=263 : stage

6. CONCLUSION

In this paper, we have introduced audience participation music that uses a distributed instrument that runs on the web browser using Web Audio API. The metaphor of collaborative music making comes from the social interaction model of online dating. In addition, we hope that this work convinces the web audio community that the cloud service adds a convenient option to support networked ensemble with minimal server-side programming. While web audio has an infinite amount of opportunities to host existing music applications, we find this piece meaningful in that it draws upon many ideas from the web both aesthetically and technologically.

7. ACKNOWLEDGMENTS

We would like to thank the students from winter class of 2015 Mobile Music Ensembles from University of Michigan to help during the rehearsals. Thanks CAPES (Brazil) for sponsoring one of the authors during the research.

8. REFERENCES

[1] J. Allison. Traversal. *WAC - 1st Web Audio Conference*, 2015.

[2] J. Alvaro and B. Barros. Computer music cloud. In S. Ystad, M. Aramaki, R. Kronland-Martinet, and K. Jensen, editors, *Exploring Music Contents*, volume 6684 of *Lecture Notes in Computer Science*, pages 163–175. Springer Berlin Heidelberg, 2011.

[3] J. L. Alvaro. Stringscore: Composing music with visual text. In *Proceedings of the Sound and Music Computer Conference*, 2012.

[4] J. L. Alvaro and B. Barros. A new cloud computing architecture for music composition. *Journal of Network and Computer Applications*, 36(1):429 – 443, 2013.

[5] B. M. J. C. M. Anand Mahadevan, Jason Freeman. Earsketch: Teaching computational music remixing in an online web audio based learning environment. *WAC - 1st Web Audio Conference*, 2015.

[6] A. Barbosa. Public sound objects: a shared environment for networked music practice on the web. *Organised Sound*, 10(03):233–242, 2005.

[7] P. Burk. Jammin'on the web-a new client/server architecture for multi-user musical performance. In *Proceedings of the International Computer Music Conference*, 2000.

[8] L. Dahl, J. Herrera, and C. Wilkerson. Tweetdreams: Making music with the audience and the world using real-time twitter data. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 272–275, 2011.

[9] A. D. de Carvalho Junior, G. Essl, and M. G. de Queiroz. Computer music through the cloud: Evaluating a cloud service for collaborative computer music applications. In *Proceedings of the International Computer Music Conference*, Denton, Texas, 2015.

[10] A. D. de Carvalho Junior, S. W. Lee, and G. Essl. Supercopair: Collaborative live coding on supercollider through the cloud. In *International Conference on Live Coding*, 2015.

[11] J. Freeman. Graph theory: interfacing audiences into the compositional process. In *Proceedings of the 7th international conference on New interfaces for musical expression*, pages 260–263. ACM, 2007.

[12] J. Freeman. Web-based collaboration, live musical performance and open-form scores. *International Journal of Performance Arts and Digital Media*, 6(2):149–170, 2010.

[13] M. Gurevich, P. Stapleton, and A. Marquez-Borbon. Style and constraint in electronic musical instruments. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2010.

[14] A. Hindle. Swarmed: Captive portals, mobile devices, and audience participation in multi-user music performance. In *Proceedings of the 13th International Conference on New Interfaces for Musical Expression*, pages 174–179, 2013.

[15] A. Hindle. Cloudorch: A portable soundcard in the cloud. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2014.

[16] A. Hindle. Orchestrating your cloud-orchestra. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2015.

[17] H. Kim. Moori: interactive audience participatory audio-visual performance. In *Proceedings of the 8th ACM conference on Creativity and cognition*, pages 437–438. ACM, 2011.

[18] T. Kita. Smartphone jam session with audience. *WAC - 1st Web Audio Conference*, 2015.

[19] S. W. Lee and G. Essl. Live coding the mobile music instrument, 2013.

[20] S. W. Lee and G. Essl. Models and opportunities for networked live coding. *Live Coding and Collaboration Symposium*, 2014.

[21] S. W. Lee and J. Freeman. echobo: A mobile music instrument designed for audience to play. 2013.

[22] J. Monschke. Building a collaborative digital audio workstation based on the web audio api. *WAC - 1st Web Audio Conference*, 2015.

[23] T. Riley. In c. Composition, 1964.

[24] S. Robaszkiewicz and N. Schnell. Soundworks–a playground for artists and developers to create collaborative mobile web performances. *WAC - 1st Web Audio Conference*, 2015.

[25] C. Roberts and T. Hollerer. Composition for conductor and audience: new uses for mobile devices in the concert hall. In *Proceedings of the 24th annual ACM symposium adjunct on User interface software and technology*, pages 65–66. ACM, 2011.

[26] C. Rogers. Web audio api. 2012.

[27] T. S. Sébastien Piquemal. Field #2. In *WAC - 1st Web Audio Conference*, 2015.

[28] B. Taylor. Pearl river (2013 2015). *WAC - 1st Web Audio Conference*, 2015.

[29] N. Weitzner, J. Freeman, Y.-L. Chen, and S. Garrett. massmobile: towards a flexible framework for large-scale participatory collaborations in live performances. *Organised Sound*, 18(01):30–42, 2013.

[30] D. Wessel and M. Wright. Problems and prospects for intimate musical control of computers. *Computer Music Journal*, 26(3):11–22, 2002.

F.11 NIME 2016 - Understanding Cloud Service in the Audience Music Performance of Crowd in C[loud]

Paper details

Title: *Understanding Cloud Service in the Audience Music Performance of Crowd in C[loud]*
Authors: Antonio Deusany de Carvalho Junior, Sang Won Lee, Georg Essl

Conference details

Title: 16th International Conference on New Interfaces for Musical Expression
Venue: Griffith University, Brisbane, Australia
Dates: July 11 to 15, 2016

Understanding Cloud Service in the Audience Participation Music Performance of *Crowd in C[loud]*

Antonio Deusany de Carvalho Junior
Universidade de São Paulo
Rua do Matão, 1010,
CEP 05508-090, São Paulo, SP, Brazil
dj@ime.usp.br

Sang Won Lee
Computer Science and Engineering
University of Michigan
2260 Hayward Ave
Ann Arbor, MI 48109-2121
snaglee@umich.edu

Georg Essl
Electrical Engineering & Computer Science and Music
University of Michigan
2260 Hayward Ave
Ann Arbor, MI 48109-2121
gessl@umich.edu

ABSTRACT

Cloud services allow musicians and developers to build audience participation software with minimal network configuration for audience and no need for server-side development. In this paper we discuss how a cloud service supported the audience participation music performance, *Crowd in C[loud]* [10], which enables audience participation on a large scale using the audience audience's smartphones. We present the detail of the cloud service technology and an analysis of the network transaction data regarding the performance. This helps us to understand the nature of cloud-based audience participation pieces based on the characteristics of a performance reality and provides cues about the technology's scalability.

Author Keywords

Audience participation, Cloud Service, Networked Music

ACM Classification

H.5.5 [Information Interfaces and Presentation] Sound and Music Computing, H..5.3 [Group and Organization Interfaces] Web-based interaction, J.5 [ARTS AND HUMANITIES] Performing arts (e.g., dance, music)

1. INTRODUCTION

It has been a long-standing endeavor to create musical performances in which the audience can easily participate. In particular, mobile smartphones have the highly desirable characteristic of already being in the possession of the audience members while offering networking and rich sensor capabilities. Golan Levin's Dialtones[13, 6] used ring-tones and wireless network dial-up to enable a concert-hall filled audience involvement piece with mobile phones. Since then much effort has been devoted to build mobile-based infrastructure to support mobile-device based audience participation such as echobo[12], massMobile[17] or Swarmed[7]. This paper aims to describe the technical realities of a recent audience participation piece called *Crowd in C[loud]* [10]. Studying cloud services in real performance settings will enable us to better understand them as they evolve. In part this paper can also be understood as advocating for the simplicity of this approach to audience participation and that some services that can be easily integrated into musical performances without requiring a deep technical background in cloud computing or virtual machines from the artist.

Our work draws upon two long-standing research and performance traditions: (1) distributed musical performances, a field of network music where people can collaborate in music making remotely, and (2) mobile music, that considers musical interaction through mobile devices. We present a three-way connection between a human, a musical instrument and the cloud server, in which the interface is built on a web page for collaborative music and the instrument communicates with the cloud server to enable social interaction among the audience. Realizing this network setup using the cloud service will require no network configuration for audience (other than visiting a web page) and no server-side programming (other than uploading the web pages) for the musicians. In addition, leveraging recent advances in web audio, we were able to distribute a complex networked musical instrument by sharing a shortened link.

In this paper we will focus on low level aspects of the technological reality of the piece, both in terms of architecture and in terms of the observed network characteristics in a public performance of the piece. For a more detailed description on the piece, please read [10] and watch a videotaped performance at `https://youtu.be/8nnrKJ4Ap0c`.

2. COLLABORATIVE NETWORK MUSIC

Network music is a computer music tradition developed to support collaboration among computer musicians [2]. The use of local networking is a rather standard piece of networked ensemble today. Interaction through a local network requires a setup effort but adapts to the performer's collaboration scheme and offers reliability as musicians know the whole network structure while defining desired settings.

One of the technical aspects of setting up network performances is addressing the hosts that will be participating. If a remote host can be correctly targeted to local network, this can be extended more globally. The conventional solution to this problem are fixed IPs. However, the necessity of a NAT for IPv4, and before the wider adoption of IPv6 around the world, the need for a public IP on the Internet limits this approach, as it is cumbersome to acquire. Some solutions mitigate this problem by choosing a single central server with a public IP. In this case, the other users will need to create a socket with the server while this server distributes data to all users. Collaborative live coding interconnected through TCP/IP (fixed IP) is a good example of extending local network to Internet and we already have

NIME'16, July 11-15, 2016, Griffith University, Brisbane, Australia.

some performances like that using Republic[1], an extension to SuperCollider, for example. However, if this solution is applied to an audience participation context, the musician will need to guide their audience to follow configuration steps. This may include downloading an app, joining a specific network and typing the IP address, which may prove difficult for casual users. Also, it might be challenging for the musician to control a large-scale crowd.

The use of online services like Twitter can save some network setup, as can be seen in Dahl's TweetDreams [3]. Roberts' Gibber [15] allows for code sharing through existing services/web browsers. Allison's NEXUS [1] leverages web services which provide interfaces for user interaction, including easy and distributed setups. Cloud computing emerged as an alternative to facilitate easier network music configuration. Hindle's CloudOrch [8] and cloud orchestra [9] are early examples that use Cloud solutions to leverage user interaction through powerful network infrastructure with less effort for the network setup. The system architecture that supported the piece *Crowd in C[loud]*, the work described in this paper, also used the Cloud, but focused on Cloud Services, as we discuss in the next section.

3. CLOUD SERVICES

Cloud services can achieve our goal of enabling audience participation in musical performances using mobile devices and provide an immersive/interactive musical environment with fast and easy set up. The "always on" paradigm and realtime interaction is now a technological reality in network settings. Much of it has been enabled by cloud computing technologies to distribute data faster through a reliable and distributed network infrastructure. As part of unlocking cloud services for network music performance we have evaluated the efficiency of a cloud service in acomputer music context and obtained good results of 83ms latency between devices in North and South America through the Pusher cloud service[2] [4]. Although this work used an Android app during the evaluations, the cloud service provides an API (application programming interface) and SDK (software development kits) for web applications that can be loaded on browsers without any system configuration or app installation. In order to facilitate remote collaboration for live coding, we created an application using the Pusher cloud service. The application developed was named SuperCopair, acts as a package for Atom.io, and explores pair programming and distributed music performance using SuperCollider [5]. Users can share SuperCollider files, edit the files together online in a collaborative session, and can also run codes locally, remotely and globally regardless of the number of connected programmers, exchanging data with the Pusher cloud service. As the communication is administrated by the cloud servers, the users only need to be connected online to have a collaborative session.

Before cloud service, the developers would have to create, setup, and start a server locally (or remotely) and develop a custom software in order to manage the user interactions and the message exchanges between connected devices. Cloud services make network setup abstract to users and distribute the data using computers clusters available in distributed data center over the globe. The available APIs offered by these services contain the functions necessary for interconnection between devices, including methods to connect, disconnect, and send messages, and also a callback to listen for received messages.

[1] `https://github.com/supercollider-quarks/Republic`
[2] Pusher Cloud Service: `http://www.pusher.com/`

Upon our evaluations on a number of different cloud services, we chose PubNub cloud service[3], which had better specification, for this performance. The advantages of PubNub, compared to Pusher, include larger message size and a greater number of messages allowed. Both services have plans that preserve the size of the messages but differ in the allowable quantity of serviced messages. While Pusher limits the messages to 10KB, PubNub accepts messages of 32KB. Another disadvantage of Pusher, especially in the context of audience participation music, is that it will discard messages if the device exceeds the rate of 10 messages per second and plans limit the number of messages sent per day. PubNub limits the quantity of messages sent per month but the limits are never throttled even on free plans for our purpose, so we can use the full service capacity during one single performance, assuming no other performance within that month. We also opted to use a paid plan for a month to guarantee that we would have technological support during the performance and a dedicated key for this application only, as opposed to using a free demo key. More details regarding PubNub Cloud Service can be found at their website: `http://www.pubnub.com/`. In the next section we will describe the piece and application created for *Crowd in C[loud]*, the performance evaluated in this paper.

4. CROWD IN C[LOUD] : ONLINE DATING THROUGH MUSIC

Crowd in C[loud] is a networked music piece composed and developed for audience participation at a music concert [10]. It draws on the idea from the piece *In C* by Terry Riley, wherein musicians (with various instruments) were guided to play pre-composed melodic fragments in C Chord [14]. In *Crowd in C[loud]*, each participant uses web browsers (typically on their smartphones) that support Web Audio API and are instructed to play a short snippet (or tune) composed by herself and by other audience members for a random amount of time. The aggregated result of each individual playing a short tune creates a heterophonic texture of chance, largely in C chord. For more detailed motivation regarding the aesthetic of the piece, see [10].

Ease of use in designing a musical interface for audience participation is one of the most significant qualities in audience participation [12]. This is especially essential to motivate people to participate in the piece with clarity and musicality. *Crowd in C[loud]* incorporates two design decisions to achieve this accessibility. First, the interaction design in the instrument is loop-based where a participant needs to place musical notes on screen and the pattern of five musical notes will create a tune that is looped indefinitely based on where the notes are. This means the user does not need to make a playing gesture constantly in order to generate tones. This nature of modifying a musical loop ensures that the instrument will generate sound so that the musician need not worry about being too sparse or silent due to low participation.

Second, the piece uses the metaphor of online dating for browsing the composed tunes of others. In the beginning of the performance, once a participant finishes the composition, he or she can browse, and play, what other audience members have composed. Browsing tunes composed by others mimics an online-dating website (such as Tinder [4]) where a user creates a personal profile and then browses other member profiles that include pictures and written descriptions about themselves. The networked instrument creates a temporary social network that lasts until the end of

[3] PubNub Cloud Service: `http://www.pubnub.com`
[4] `www.gotinder.com`

the performance where each tune is a musical profile of a participant. In addition, the collection of each tune composed by individuals serves as musical phrases found in Riley's In C.

The metaphor of creating and browsing online profiles creates a set of states a participant can be in during the performance. The musical instrument for a user can be in one of five different states: NAME, EDIT, WAIT, CHECK, and MINGLE. Each state is used to design different interfaces and different modes of social interaction: 1) NAME is the initial state where a user determines the screen-name for participation. 2) EDIT is the state in which a user can compose and modify the looped tune to create a profile. 3) WAIT is a transient state that involves waiting for data response of tunes (typically composed by other audience members) 4) CHECK is the state where one can browse (and play) someone else's tune and 5) lastly, MINGLE is the state where a participant can play two tunes at the same time, which is a metaphor of conversation, off-line meeting, or being a match. The times in which a participant makes a transition from one state to another are the points when a participant's smartphones request data from the cloud service.

In *Crowd in C[loud]*, the only sounds that constitute the piece are coming from the audience seats, from speakers of the audience's smartphones. There is a performer on stage who runs a performer's interface, which serve as a scoreboard-like visualization in which stats of each profile is displayed (e.g. the number of likes). In fact the computer that the performer runs acts as a server where all the audience's tunes are stored and stats are calculated. Whenever a participant modifies a pattern or browses audience member's tune, the audience's interface request data from the performer and waits for its response, which would come via the cloud server from the performer's web page loaded at the laptop on stage. Additionally, the performer can orchestrate the crowd by live coding in Javascript. The on-the-fly code is distributed to all connected devices and is used to change the property of the instrument that audience play, as seen in [11]. For example, a performer can change chord and scale of the instrument so aggregated outcome of tunes can make chord progression over time. Therefore all devices are connected and will transmit data to the performer's laptop. In the following sections, utilization of cloud service for data transmissions will be discussed.

5. PERFORMANCE STRUCTURE FOR AUDIENCE INTERACTION

The performance structure was inspired by other networked pieces that presented a centralized network with a server for receiving and sending information [16]. However, our choice of using the cloud service set us free from the burden of developing a custom server application, which is replaced with a single web page written in HTML and Javascript.This will be a useful setup for artists who want to write a network music piece who do not have a networking background but can write an interactive web page. As its name suggests, PubNub cloud service permits easy interconnection between users through channels using publish-subscribe (or pub-sub paradigm). An application (or device) can subscribe to a channel and receive every notification that is published to the channel. For example to broadcast a message to all devices, a device can publish a message to a channel that every device is subscribed to. This provides a convenient abstraction that is robust against changes in network or end-user device and allows dynamic reconfiguration of participation. The push(or publish) notifications paradigm also

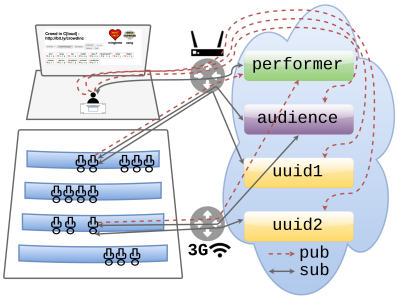


Figure 1: Performance structure. The left side depicts the performance space. On the right side, a rectangle represent a channel, solid lines show publishing and dotted lines show subscription.

Price/month	Free	$15	$49	$125	$399
Daily devices(max)	20	100	3k	10k	25k
# of msg per month	1 million	Millions(underspecified)			
Additional messages	-	Up to $5/M			

Table 1: PubNub pricing plan(April 2015)

makes performances more robust against technical disruptions such as disconnection and delays.

The left side of Figure 1 presents the performance setup at the concert hall. There were two kinds of web pages running during the performance, one for the performer and another for the audience. The performer sat center-stage and his application (a web page) was project onto a screen for the participants seated in the audience. They were instructed to visit their application that has the musical instrument on. The performance hall had wireless network available from the university and had a good reception of mobile network connectivity such as 3G.

A representation of the cloud service with the channels we used is on the right side of the Figure 1. In our performance we designed three types of channels following the PubNub API for web application: a performer channel, an audience channel and individual channels equaling the participant number. The performer application is subscribed to the performer channel to receive messages sent from audience member's devices. Once a participant visits, it requests a performer's application to create an individual channel by publishing a message to the performer channel. The individual channel is then created to respond to an audience application's request, typically for the audience to retrieve a tune composed by others. When the page is loaded, every device is given a universal unique identifier (UUID) from the cloud service API, which is used to create an individual channel. Lastly, all the audience members are subscribed to the audience channel. This channel is used for crowd control: to send textual instructions, to live-code the mobile instrument, to start/end the performance and to troubleshoot the performance when it goes wrong (silence, refreshing the page).

The service plans available from PubNub at the time of the performance were divided into free and paid plans, and the price is described at Table 1. The number of messages was not an issue for the performance context (even with the free plan), because we would use the application only during

rehearsals and on the performance day. However, the free plan limits the maximum number of daily devices to 20. As we could not predetermine the number of turnout for the performance, we could limit the number by the capacity of the concert hall (450), which led us to choose the third plan option in the assumption that there may be more than 100 devices.

PubNub cloud service provides an extensive API and SDKs, and the service can be easily configured with a few functions. Initially, the application needs to get an UUID and initialize the PubNub object including information regarding the account: the publish key and the subscribe key. After that, the device can publish messages to any channel as long as the channel name is known (e.g. "performer", "audience"). When subscribing to a channel, it is also necessary to define a callback function that will typically parse the received messages. The basic function to set up a publish-subscribe mechanism is presented on Listing 1. The code of two applications are available in the open-source repository at:

`https://github.com/panavrin/tindermusic`.

The structure of the code and performance is similar to other performances that follow the same paradigm of audience participation, but our solution avoids any server implementation or any intercommunication manipulation, thanks to the advantages of the cloud service. The actual physical machines are abstracted and run in the background to realize the performance. The audio generated from that application was based entirely on Web Audio and could run on any compatible device, including many mobile devices that can run a web browser that supports Web Audio. In the following section, the performance is evaluated in terms of the network message transactions.

```
// Request an UUID
var my_id = PUBNUB.uuid();
// Initialize with Publish & Subscribe Keys
var pubnub = PUBNUB.init({
 publish_key: publishKey,
 subscribe_key: subscribeKey,
 uuid: my_id,
});

// Subscribe to a channel
pubnub.subscribe({
 channel: my_id + ",audience",
 message: parseMessage, // callback for msg
 error: function (error) {
  // Handle error here
 },
 heartbeat: 15
});

// Parse received message
function parseMessage( message ) {
 if (typeof message.type !== 'undefined'){
  if ( message.type == "create-response"){
   // Do something
  }
 } else if ...
}

// Publish a message to a channel
pubnub.publish({
 channel: "performer",
 message: {"type":"create",
  "my_id":my_id,
  "nickname": strScreenName},
 error : function(m) {
  // Handle error here
 }
});
```

Listing 1: Example of JavaScript code from PubNub API presented at audience page

6. PERFORMANCE RESULTS

In April, 2015, "*Crowd in C[loud]*" was premiered at the annual concert of the Mobile Phone Ensemble, University of Michigan. Our performance lasted for 6 minutes with the sole sound from audience creations, proceeded by a performer's introduction explaining how to participate. We developed applications to log transmitted messages among connected devices during the piece, using a computer located on site and a remote server that listens to the cloud service. The two logs were compared and were identical other than the timestamps. Given the remote server was physically the farthest machine (California) from the other machines at the site of the performance hall (Michigan) and the cloud data center (Northern Virginia), it is reasonable to claim that message loss was unlikely.

The messages logged were limited to the performer and the audience channels as they were predefined and the messages are anonymous. Individual channels of audience members that were created on the fly were not logged for this study. As the purpose of this study was to understand network traffic of the performance, no attempts were made to provide logging details that would have allowed identification of individual user behavior. Rather user numbers are used to indicate overall load and temporal patterns on the network. The time window in which we analyzed the messages is defined by the time that the performer went live (or pressed the "go live" button) and lasted for the actual performance (six minutes).

The number of UUIDs created by PubNub during the performance was 184, indicating that the audience interface page was visited 184 times (including page refreshing). On the other hand, we had only 76 different screen names entered during the performance, and 69 of them send/receive messages for the actual participation. The high number of UUIDs compared to the number of screen names is because we did not restore previous sessions of UUIDs when refreshing the screen and the performer sent a message that refreshes the page right before the performance. Another discrepancy in the number of screen names (76,69) may be caused by the incompatible devices or voluntary disconnection from the participation.

Figure 2 presents the duration of individual user's participation during the performance. Each horizontal line is drawn from the time a participant submits the first message to the performer channel until the system observes the last message. Most lines starts approximately one minute after the performance started, indicating that initially participants spend time in composing their musical profiles (or composing tunes). The short line indicates that the user was active for a short amount of time. This reflects the cases in which a user refreshed the page due to the delay from the cloud and created another screen name after refreshing the page. Therefore, multiple lines from a single participant may exist.

Based on Figure 2, the total number of users connected at the same time is calculated in the Figure 3. The maximum value in the graph, which indicates the maximum number of

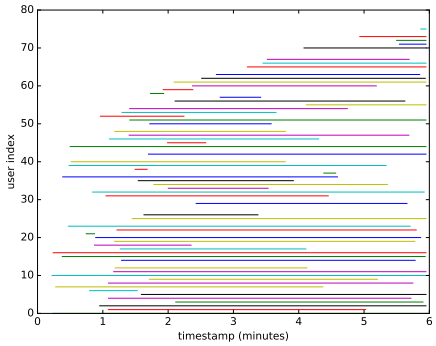


Figure 2: Participation of the audience during performance

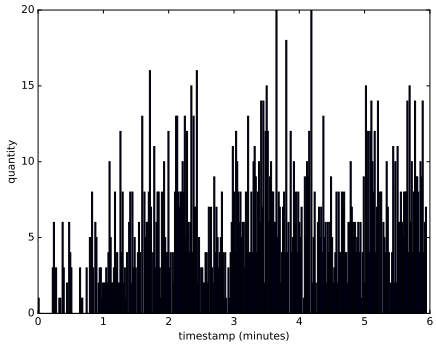


Figure 4: Histogram of the number of messages in each second during the performance (Estimated)

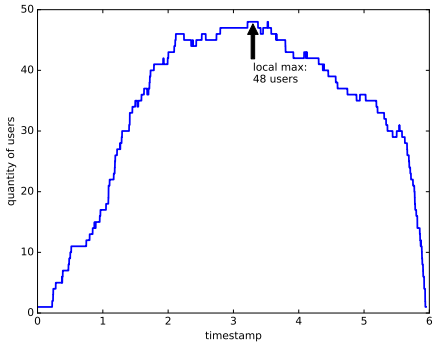


Figure 3: Number of users connected during the performance

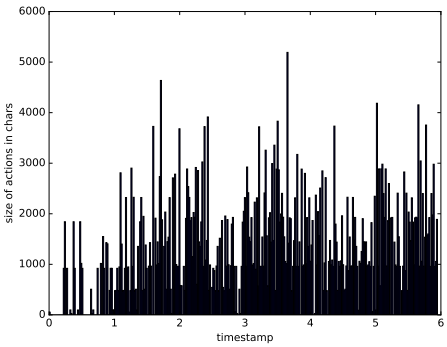


Figure 5: Size in bytes of the messages in each second during the performance (Estimated)

users online at a specific moment during the performance, was 48. This reflect the practical number of participants at the performance based on our observation of the number of turnouts at the concert.

One of the significant aspects of the analysis is to develop metrics so that we can estimate maximum network traffic per second (bytes/sec) in order to evaluate the feasibility of the network setup given the scale of the performance. This will be a useful measure to secure the performance in terms of network configuration based on the estimated number of attendance and the networked interaction scheme of the musical application. Based on the log data available, all the transaction messages (that are not logged) are estimated according to the PubNub API and the message protocol that we defined. The size and number of estimated messages are close to the reality because the missing messages are in response to the request messages (that the logging application kept track of) and we can calculate the fairly accurate size of each message type based on the protocol regardless of its value.

The Figure 4 and 5 show two different metrics with which we can infer the feasibility of the network configuration. Figure 4 shows the number of transmitted messages per second, and Figure 5 presents the number of bytes in the messages sent per second during the performance. In both graphs, the peak of the graphs matches the time window that includes the peak in Figure 3. The number of messages matters because the cloud service keeps track of the number of messages and can limit the functionality by the monthly plan of choice. In the meantime, the metric in Figure 5 helps us estimate the bottleneck step in the network pipeline. While these two graphs can vary as the number of bytes per message differs based on its type (25 to 470 bytes), two graphs show similar patterns. Given that the cloud service allowed messages up to 32 kBytes, the message that the program required was at most 470 bytes(= 0.00047 kBytes). In addition, most network devices consider the maximum transmission unit (MTU) as 1500 bytes and divide messages into different packet, which indicates that all messages would be sent in one packet.

In total, we had 86961 bytes received by the performer, 370,630 bytes sent by the performer, and 457,591 bytes in total, during the 6 minute-performance. In terms of bandwidth, we can assure that the total data exchanged during the whole performance including entire participants did not exceed 1Mb. This information can be useful for the audience to alleviate their concern with the data usage caused by participating in the performance if they were using limited mobile data plan. Rather, loading the web interface page in the beginning constituted the largest share of data downloaded at 1.5MB. Based on our observation, the performance would not have been affected even if we chose PubNub's cheapest service plan in terms of bandwidth. Unsurprisingly, the device with the heaviest network traffic was

the performer's laptop (at under 0.5 Mb for the whole performance). However, since any delay (either computational or network) in the performer's laptop would impact all connected devices, it is recommended to use wired connection to route the data separately as well as optimize the code that is running that may contribute to the delay.

7. DISCUSSION AND CONCLUSIONS

In this paper we presented utilization of a cloud service for audience participation in musical performances. The amount of network configuration both for the developers and audience members was minimal. The network configuration using the cloud service allowed performer to mediate the orchestration of the piece and let audience members participate and collaborate with the social interaction metaphor. Our analyses on the performance log show the cloud service had more than enough capacity to service the performance and gives us some information regarding what we should consider for upcoming performances. We discovered that the bandwidth required for this performance is low given the specific interaction scheme (browsing musical profiles) among audience members. This minimize our concern of failure for heavy traffic and audience concern of data usage consumption for participation. This is particularly significant given that audience participation cannot be easily be rehearsed in realistic setting upfront.

Regarding the cloud service plans, we note that the number of messages can hardly limit the network usages of the performance, even if we would have performed everyday during the whole month. However, the number of synchronously (daily) connected devices could have been the limit and actually could have been the case if we used the free plan given our miss on restoring session of UUIDs. The dress rehearsal on the same day should have consumed a certain number of devices. In general, we can conclude that the cloud service was stable for the performance with a strong evidence of no message loss.

For the future, we plan to further investigate and improve the performance practice. First, the log application will be modified to complete all the metrics that we had to estimate in this work.Not only will this let us measure the actual network traffic but we will also be able to analyze the user experience during the performance. For example, we will be able to detect network delay for individual and dropout when a user refreshes the page due to a delay in information retrieval. Second, we are in the process of enhancing the user experience in case of contingency. Given that all the relevant code to run the musical instrument is already downloaded locally, the interactivity should have been continued while the application waits for a response from a server. Lastly, we are interested in understanding how collaboration among audience members changed their experience with the performance. Learning whether the social aspects of the instrument help the audience sustain their interest in participation is important to garner further understanding in prolonged audience engagement.

8. ACKNOWLEDGMENTS

We would like to thank the students from winter class of Mobile Music Ensembles from University of Michigan. Thanks for CAPES(Brazil) funding during the research and Emma Planet support on paper proofreading.

9. REFERENCES

[1] J. T. Allison, Y. Oh, and B. Taylor. Nexus: Collaborative performance for the masses, handling instrument interface distribution through the web. In *Proceedings of New Interfaces for Musical Expression*, 2013.

[2] Á. Barbosa. Displaced soundscapes: A survey of network systems for music and sonic art creation. *Leonardo Music Journal*, 13:53–59, 2003.

[3] L. Dahl, J. Herrera, and C. Wilkerson. Tweetdreams: Making music with the audience and the world using real-time twitter data. Citeseer.

[4] A. D. de Carvalho Junior, G. Essl, and M. G. de Queiroz. Computer music through the cloud: Evaluating a cloud service for collaborative computer music applications. In *Proceedings of the International Computer Music Conference*, Denton, Texas, 2015.

[5] A. D. de Carvalho Junior, S. W. Lee, and G. Essl. Supercopair: Collaborative live coding on supercollider through the cloud. In *International Conference on Live Coding*, 2015.

[6] A. de Souza e Silva. Art by Telephone: from static to mobile interfaces. In Lanfranco Aceti, editor, *Leonardo Electronic Almanac*, volume 20. Leonardo On-line, 2004.

[7] A. Hindle. Swarmed: Captive portals, mobile devices, and audience participation in multi-user music performance. In *Proceedings of the 13th International Conference on New Interfaces for Musical Expression*, pages 174–179, 2013.

[8] A. Hindle. Cloudorch: A portable soundcard in the cloud. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, London, United Kingdom, 2014.

[9] A. Hindle. Orchestrating your cloud-orchestra. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2015.

[10] S. W. Lee, A. D. de Carvalho Junior, and G. Essl. Crowd in c[loud]: Audience participation music with online dating metaphor using cloud service. *In Proceedings of Web Audio Conference*, 2016.

[11] S. W. Lee and G. Essl. Live coding the mobile music instrument. In *Proceedings of New Interfaces for Musical Expression (NIME)*, Daejeon, South Korea, 2013.

[12] S. W. Lee and J. Freeman. echobo: A mobile music instrument designed for audience to play. *Ann Arbor*, 1001:48109–2121.

[13] G. Levin. Dialtones - A telesymphony, September 2001, 2001.

[14] T. Riley. In c. Composition, 1964.

[15] C. Roberts and J. Kuchera-Morin. Gibber: Live coding audio in the browser. In *Proceedings of the International Computer Music Conference (ICMC)*, Ljubljana, Slovenia, 2012.

[16] G. Weinberg. Interconnected Musical Networks: Toward a Theoretical Framework. *Computer Music Journal*, 29:23–39, Jun 2005.

[17] N. Weitzner, J. Freeman, Y.-L. Chen, and S. Garrett. massmobile: towards a flexible framework for large-scale participatory collaborations in live performances. *Organised Sound*, 18(01):30–42, 2013.

F.12 WAC 2017 - Open band: Audience Creative Participation Using Audio Synthesis

Paper details

Title: *Open band: Audience Creative Participation Using Audio Synthesis*
Authors: Ariane Stolfi, Fábio Goródscy, Antonio Deusany de Carvalho Junior, Fernando Iazzetta, Mathieu Barthet

Conference details

Title: 3rd Web Audio Conference (WAC)
Venue: Centre for Digital Music, Queen Mary University of London, London, UK
Dates: August 21 to 23, 2017

Open band: Audience Creative Participation Using Web Audio Synthesis

Ariane Stolfi
Universidade de São Paulo
05508-020 São Paulo
arianestolfi@gmail.com

Mathieu Barthet
Queen Mary University of London
E14NS Mile End Road
m.barthet@qmul.ac.uk

Fábio Goródscy
Universidade de São Paulo
05508-090 São Paulo
fabiogorodscy@gmail.com

Antonio Deusany de Carvalho Junior
Universidade de São Paulo
05508-090 São Paulo
dj@ime.usp.br

Fernando Iazzetta
Universidade de São Paulo
05508-020 São Paulo
iazzetta@usp.br

ABSTRACT

This work investigates a web-based open environment enabling collaborative music experiences. We propose an artifact, Open Band, which enables collective sound dialogues in a web "agora", blurring the limits between audience and performers. The systems relies on a multi-user chat system where textual inputs are translated to sounds. We depart from individual music playing experiences in favor of creative participation in networked music making. A previous implementation associated typed letters to pre-composed samples. We present and discuss in this paper a novel instance of our system which operates using Web Audio synthesis.

1. INTRODUCTION

Recent advances in web audio and open source technologies provide new grounds to reconfigure interactive audio art. In this work we propose a web-based platform for musical creative participation bridging the gap between audience and performers. We present an implementation, Open Band, where participants can play sounds in response to textual messages entered in an on-line chat. Our implementation is based on web audio technologies since they support the development of accessible audio interfaces [25], are ubiquitous and easy to distribute [18]. Furthermore they can easily be applied to design interactive systems, independent of any extra software installation [31].

We are interested in creating "open" works, in the sense advanced by Umberto Eco [9], were the final arrangement of the piece is given to publics to decide. We use music as our domain as music is known for its social functions of cohesion and communication [16]. Our web-based instrument has been used to create experimental participatory music performances where everyone in the audience can join through its cell phone, tablet or laptop, and can playing through typing.

The first version that we developed converted letters into sounds using audio samples. In the version presented in this paper, we are building a piece entirely based on web audio synthesis to reduce bandwidth requirements for the audience participation, and also to experiments with possibilities given by the web audio API.

Web Audio Conference WAC-2017, August 21–23, 2017, London, UK.

2. RELATED WORKS

Participatory art and Eco's Open Works.

Contemporary Western music performances are predominantly presentational, with performers preparing and providing music for another group, the audience [27]. In contrast to presentational performances, which intrinsically create a divide between audience and performers, participatory performances [15] seek to blur the boundaries between audiences and performers by giving equal role to participants, building up on the notion of "communitas" (unstructured communities in which all members are equal allowing them to share a common experience). Such aspiration from participatory art corroborates Eco's conceptions of "open works", which, according to Robey in [8], require of the public *"a much greater degree of collaboration and personal involvement than was ever required by the traditional art of the past."*. In Open Works, it is *"the artist's decision to leave the arrangement of some of their constituents either to the public or to chance, thus giving them not a single definitive order but a multiplicity of possible orders"*. These motivations interestingly resonate with the contemporary perception of audience's role changing from primarily passive to one "co-creating values" [23], with audiences who increasingly want to "shape" their own experiences. Another important rationale of participatory art is to improve audience cognitive engagement in performance through an active form spectatorship involving a physical engagement in the performance [15].

Participatory music performances with Web technologies.

Human-computer interaction (HCI) and communication technologies provide great potential to facilitate participatory art forms in our digital age as they can be used to mediate and transform creative information remotely, in

(quasi) real-time and scalable ways. Within the field of digital participatory art, several authors investigated how to create participatory music performances using web technologies (see [30] for a review of some of them). Web applications for participatory performances have been proposed e.g. in [28, 10, 32, 3, 17], each of these studies manifesting specific artistic and audience creative agency models. The massMobile framework [29] was used to let audiences control stage lighting effects or create musical loops from instructions provided through a graphical user interface (GUI) on their smartphones. Mood Conductor [10] also relies on the use of smartphones to enable audience members to conduct performers in terms of emotional directions. In Open Symphony [32, 30], audience members can generate live graphic scores interpreted by performers by voting for music playing modes on their smartphones. In this work and others, such as A.bel [3] and the CoSiMa framework [17], smartphones are employed both as a sound control and diffusion interface leveraging the loudspeakers embedded in the devices.

3. OPEN BAND DESIGN

During the development of this new version we opted to evaluate some ready-made solutions for web audio synthesis like Gibber [24], Waax [2], and meSpeak.js [1]. These solutions are able to facilitate the use of web audio technology and other browser facilities, but also create barriers and sometimes limitations that weren't imposed by pure web audio. Furthermore, sometimes frameworks could offer more than what was required by the application, which can lead to more confusion during development. We tried to use, meSpeak.js as a simple and interesting text-to-speech library but the resulting sound was not musically satisfactory, and we couldn't make messages play one on top of each other, due to limitations of the framework, so we choose to focus on pure web audio for sound programming.

We also decided to follow some aesthetic concepts based on type design [26] to propose "an audio typograph", and also to experiment with new kind of music, avoiding traditional chords and scales. These concepts surrounded the whole development of this current version of Open Band and had a great impact in the new result obtained.

3.1 Genesis

The phonetic alphabet was as pointed by McLuhan [21], a fundamental technology for the development of our culture, being easy to learn and adjustable to many languages. Here we are making use of the alphabet as a technology to produce experimental music. Since text messages are one of the main forms of communication on smartphones today, we choose to experiment possibilities to experiment with type as input to produce interactive music.

Interfaces using typing as input are used in various applications such as live coding [4], audience participation pieces, to receive feedback [2], to get user data as a source for algorithmic composition [5], or using keyboard as music controllers [11]. In this project, the interaction process happens through an open multi-user chat interface (see Figure 4), where there is no distinction between users. No one can tell who is playing what and each message sent to the chat is loaded into a chat room and is played back as a sequence of sounds. In an inter-semiotic translation [22], defined by Roman Jakobson as 'transmutation of signs', [6], we decode the text messages in musical information, and all messages are played as they are sent to the chat. In this particular version, we are also using a concept of audio typography as base to this inter-semiotic translation, to decode letters into sounds, building spectral designed "letters" made of predetermined sound blocks. Trough this path, we are abstracting the aural aspects of the typed characters and grabbing only the shape of the letters as information for the sound synthesis.

3.2 Audio Typography

The effect of typography on written texts acted as a motivation in the present artifact to experiment with ways in which timbre could affect the sound produced by musicians. We use additive and noise synthesis to sonify "drawings" of the shapes of the letters.

3.2.1 Metafont

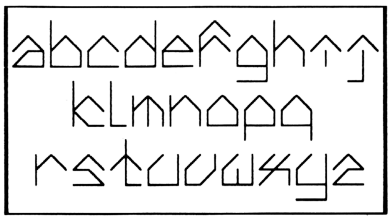


Figure 1: One of the modular alphabets proposed by Douglas Hoefstader in the book Metamagical Themas, page 90

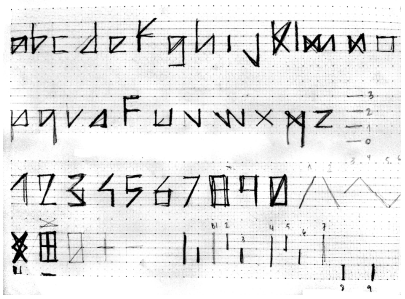


Figure 2: Sketch for the modular font to be base for audio synthesis.

Donald Knuth, a computer scientist and artist who worked on typesetting issues, developed the concept of a "meta-font", a typeface that was not statically drawn, but that could be changed trough typographical parameters[7]. Following the rules of meta-fonts, many different typefaces could be generated simply by varying the type parameters. As says Douglas Hoefstader in [14], *"Knuth's purpose is not to give the ultimate parametrization of the letters of the alphabet (indeed, I suspect that he would be the first to laugh at the very notion), but to allow a user to make "knobbed letters" – we could call them letter schemas. This means*

[1] meSpeak.js website: http://www.masswerk.at/mespeak/

[2] FLOrchestra at CMMR http://www.crisap.org/event/in-transglasphone/

that you can choose for yourself what the variable aspects of a letter are, and then, with Metafont's aid, you can easily construct knobs that allow those aspects to vary.".

> Knuth's purpose is not to give the ultimate parametrization of the letters of the alphabet (indeed, I suspect that he would be the first to laugh at the very notion), but to allow a user to make "knobbed letters" – we could call them letter schemas. This means that you can choose for yourself what the variable aspects of a letter are, and then, with Metafont's aid, you can easily construct knobs that allow those aspects to vary [14].

We used the basic parameters defined by Knuth [14] such as x-height, baseline, height of ascendants and descendants, as measures to establish frequencies for tone sounds (one low, one at the baseline, one at the x-height an one at the uppercase line of each letter). The horizontal parameters of the type are translated into temporal measures, to determine duration of the sound events and intervals between the letters.

A meta-font is a complex font with several typographic parameters. In this work, we are using a simpler analogy by proposing a modular type, similar to the ones Douglas Hoefstader [14] proposes in his book "Metamagical Themas", based on simple grid structures (see Figure 1). We propose a model that can work only with a few building blocks (see Figure2 in which they are sketched). The line of the letters was mapped into two basic types of functions: noise synthesis for the verticals and solenoids for the horizontal lines and glissandi, as defined by the composer project in Figure 3.

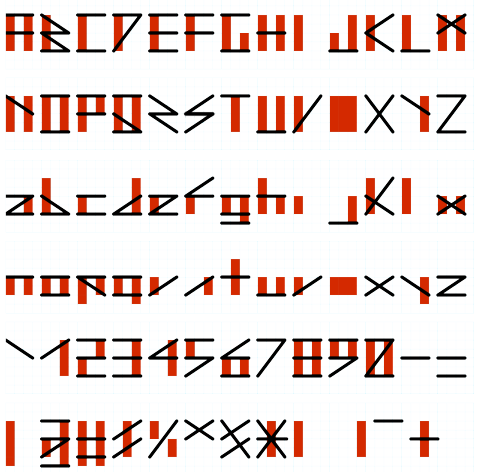


Figure 3: Project for audio typography, made of blocks of noise, sine waves and glissandi. The red blocks represent noise and the black lines solenoids and glissandi.

3.3 Project Development

This project started off as open source software in 2016. Although its applications are oriented towards the artistic domain, we have been using agile design methodologies principles generally used for products. We namely use the Lean Ux [20] methodology that includes early user evaluation, collaborative cross-functional design, solving use case problems, and nimble design.

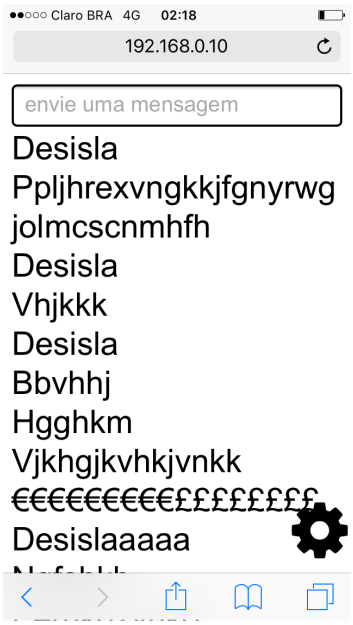


Figure 4: Interface of the chat on mobile device

3.3.1 *Front End*

The interaction takes place trough a webchat interface an illustration of which is given in Figure 4. When users enter the site, they can type messages in a text box, with no distinction between past and present users. Once the messages are received they are played as a sequence of preprogrammed sounds. The composer and programmers decided to keep the interface is minimalistic for aesthetic reasons and in an attempt to follow a brutalist approach as it stands in architecture.

3.3.2 *Special Commands*

Besides sounds generated from people's interactions, there are also commands that can be live coded. Those commands are not shown in the public interface, for the conductor maintain certain level of control on the performances. They can change the audio frequencies, the global volumes or turn off the sound, for example. The commands follow a command line like program paradigm, which means that there are special words and characters that will run special actions, like programs. This kind of feature is also present in many of modern chat-like applications, so we expect this to be intuitive for users. Also, we have an administration interface with pre-programmed frequencies and intervals, that we can use to control sound flow and rhythm in real time.

3.3.3 *Network Architecture*

The project was built having two server components that are completely independent, one is a typical web server which is responsible to serve web content such as HTML, JavaScript and audio files, based on the Web Audio API [1] and the other is responsible for making messages deliverable (see Figure 5). This provides flexibility to the system, as it allows the messaging server to be replaced by any framework that supplies what is required by the application.

3.3.4 *Back-end*

In the current implementation, the messaging server is written in Ruby, making use of the Puma, Sinatra and Faye frameworks, as well as WebSockets for the server/browser communication. The server [3] tries to be easily reproducible on Linux platforms and is released open source. The messaging technology plays a main role in such application and early versions of the project frameworks like PubNub and Peer.js have been tested. The decision of keeping a Ruby server was made having in mind that it was important being able to make the performances even without any Internet access, what couldn't be made with these frameworks.

3.3.5 *Front-end*

In the first version, all the sounds are played as samples. In this case we have one sample for each ASCII character, and the mp3 files are named each with its ASCII number. This one-to-one mapping was chosen to make the system more recognizable by its players, as results can always be expected. Sounds are played one after another, as indicated by the sequence of the letters on the messages. The samples are rhythmically played through values that are determined by the code and that can be changed through special commands in real time. In the proposed version, sounds are all synthesized in the browser. This way there is no need for transferring files at the beginning of a user's participation. The stream of data is reduced and more parameters can be changed during a live performance.

3.3.6 *Performance Set-up*

The performances are designed to be made in any space that can fit a laptop and a network hub. Participants can join the network with their devices and connect to the the application in a web browser. As nowadays' web browsers are ubiquitous, the only limitation for connecting to the application is the device being able to join the network through WiFi. This way, the requirements are quite minimal for the audience. We usually use a projector to show the messages for everybody, so even who is not connected to the application can enjoy the action.

A regular laptop should be able to run the application without any issues. The last performances were made on a laptop with AMD A4-5000 processor and 4GB memory. Any devices that can install Ruby gems should be able to host the server but it was created and developed for Linux.

3.3.7 *Audio Synthesis*

In the first version, all the audio processing was made through sampling, which caused a long time for downloading the samples from the server when the user entered the website. Also, the samples had to be cut one by one, resulting in a large amount of time to prepare material for playing.

[3]Server code: https://github.com/fabiogoro/bandaserver

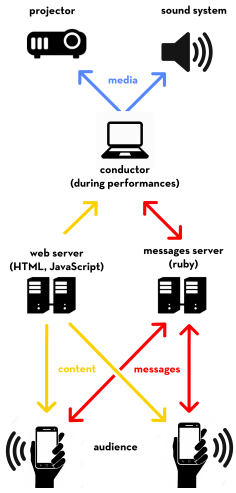


Figure 5: Servers Schema

The new version relies entirely on web audio synthesis, using the Web Audio API to generate sounds on users' devices. There could be many forms to do the mapping between letters and sounds, such as note/keyboard association, that used in many applications such as DAW software, or even speech synthesis mechanisms. The present artifact draws on the idea of "audio typography". We are drawing spectral "letters" made of modules of audio functions, that work as temporal semiotic Units (TSU) [13]; such as noise blocks, sine waves and glissandi. Now there are two types of basic functions that work as building blocks, as shown in Figures 6 and 7, one that uses FFT synthesis to build the blocks of noise based on variable values of tempo and base frequencies, using the framework Noisy.js [4], witch was written specially for this project, but can be used to other web-based music projects in the future.

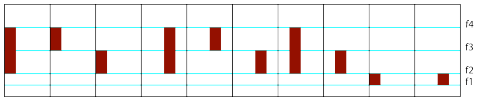


Figure 6: Functions for playing vertical blocks with noise synthesis

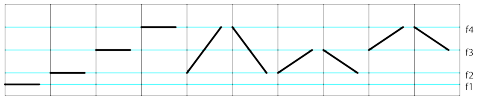


Figure 7: Functions for playing horizontal and diagonal lines with oscillators

Sine function has one oscillator buffer that can have static frequencies or linearly ramping frequencies. This creates diagonal and horizontal lines in the spectrum. Then, frequencies are defined as the bottom and top limit, some steps are defined in the middle. The length of spectral letters are as

[4]Source code: https://github.com/fabiogoro/noisy

well defined. By the end, letters are translated as a group of functions, and the result is played one after another. We built 20 predetermined functions to correspond at each "type block", and each letter plays as many functions as are determined by the typeface project. Thus, we created a system of mapping letters to the functions. Every output node goes through an ADSR class [19] that creates a more natural attack/release, and also goes through a gain node, that can be changed during a performance. On Figure 8 is shown a spectrogram generated by the application with the results obtained.

Figure 8: Spectrograms generated by the application

3.3.8 Challenges

As every letter from every message is translated into a group of playing Web Audio nodes, the number of oscillators that plays simultaneously can grow really quickly depending on the number of participants sending messages. As this application was intended to play in many different mobile devices, this led to the necessity of limiting the creation of nodes, as this could take more processing than some devices could take, and in some cases could make the device stop playing, which is undesirable behavior for such application. We have also met differences of implementations from web audio synthesis in different browsers which led to unexpected behavior between different executions. As an example, the way time is used in linearRampToValue in Safari and Firefox browsers is different from Chrome, and iOS devices request special actions for playing sounds in browsers. For now the system is working correctly on recent versions of Google Chrome, Mozilla Firefox, Safari and Opera browsers.

4. PERFORMANCES EVALUATION

We have presented the first version of Open Band as performance running the server in a local computer and a closed WiFi network, for reducing latency and keep the public together in a kind of web"agora". It has been performed in different occasions, in concerts and private spaces, most of them with a projection of e online chat on a shared screen. Depending on the audience's profile, the exchanges between participants could go different way. We observed that in musical contexts, the audience keeps more time experiencing with the samples and rhythm, typing meaningless phrases, and when the public is younger, the participants tend to play with the ability to speak anonymously. As the conversation "warms" up, layers of sounds are overlapped, turning the rhythm into something more frenetic.

Like in a real conversation, if people don't take time to listen to the others, the sound becomes more difficult to distinguish. We received a lot of feedback from the public on the chat, with people asking things about the project and supporting it.

One problem of the previous version was that been based on heavy volume of data from samples, we had always to create a local network and even with that we had a long time of download before playing. With the present version relying only on web audio synthesis, the size of the project reduced dramatically — from 60 megabytes to kilobytes — and also the downloading time when compared with the previous version. This new aspect of the Open Band facilitates users to participate using 4G data plan as the data consumption is now comprised of text messages only as no samples files are downloaded.

It is important to emphasize that the browser used by the participants can make a real difference on the performance itself. Most current browsers do enable web audio synthesis, however they are incompatible with many parameters. That said, the participants are recommended to use Google Chrome and take advantage of its full compatibility with current web audio standards.

Figure 9: people interacting within the piece

5. CONCLUSIONS AND FUTURE WORK

Free interaction and easiness to use bring a high degree of fun to the activity of making music with web audio applications, as we have seen in some of performances. It does make play music closer to the sense of playing that is also related to games, but, in this case, maintaining a socio-political standpoint, offering freedom of speech, and allowing individuals to interact without been pointed at, granting people to express personal ideologies without constraints. Besides this, since the messages are decoded into music, there is also a possibility of communication beyond symbolic written language, with the sharing of purely sound phrases, without any literal sense.

There are now two working versions of the project, one with fixed collections of samples, and the other with web audio synthesis, both of them with preprogrammed sounds for each letter. We seek in the future to expand it to be a framework to be used in different musical contexts. One of our goals is to integrate the project with the audio Commons API [12] to foster media re-purposing, creating an interface for uploading and mapping sounds to the letters. For the web audio synthesis version, we aim to develop form to control the base frequencies and rhythm trough gestural interfaces, to enhance the musical possibilities given to the audience.

6. REFERENCES

[1] P. Adenot, C. Wilson, and C. Rogers. Web audio api. https://www.w3.org/TR/webaudio/, 2017. Accessed March, 2017.

[2] H. Choi and J. Berger. Waax: Web audio API extension. In *Proceedings of the International Conference on New Interfaces for Musical Expression*,

pages 499–502, Daejeon, Republic of Korea, May 2013. Graduate School of Culture Technology, KAIST.

[3] A. Clément, F. Ribeiro, R. Rodrigues, and R. Penha. Bridging the gap between performers and the audience using networked smartphones: the a.bel system. In *Proceedings of International Conference on Live Interfaces*, 2016.

[4] N. Collins, A. McLean, J. Rohrhuber, and A. Ward. Live coding in laptop performance. *Organised Sound*, 8(03):321–330, 2003.

[5] L. Dahl, J. Herrera, and C. Wilkerson. Tweetdreams : Making music with the audience and the world using real-time twitter data. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 272–275, Oslo, Norway, 2011.

[6] Daniella Aquiar and João Queiroz. Towrds a Model of Semiotic translation. *The Internationa Journal of Arts in Society*, 4, 2009.

[7] Donald E. Knuth. The Concept of Meta-Font. *Visble Language*, XVI:3–27, 1982.

[8] U. Eco. *The Open Work (Translated by Anna Cancogni With an Introduction by David Robey)*. Harvard University Press, Cambridge, Massachusetts, USA, 1989.

[9] U. Eco. The poetics of the open work. 1962. *Participation: Documents of Contemporary Art*, pages 20–40, 2006.

[10] G. Fazekas, M. Barthet, and M. B. Sandler. The Mood Conductor system: Audience and performer interaction using mobile technology and emotion cues. In *Proc. of the Int. Symposium on Computer Music Multidisciplinary Research (CMMR)*, 2013.

[11] R. Fiebrink, G. Wang, and P. R. Cook. Don't forget the laptop : Using native input capabilities for expressive musical control. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 164–167, New York City, NY, United States, 2007.

[12] F. e. a. Font. Audio commons: Bringing creative commons audio content to the creative industries. In *Proceedings of Audio Engineering Society Conference: 61st International Conference: Audio for Games*. Audio Engineering Society, 2016.

[13] X. Hautbois. Temporal semiotic units (tsus), a very short introduction. *MiM.*, 29, 2013.

[14] D. Hofstadter. *Metamagical Themas: Questing For The Essence Of Mind And Pattern*. Basic Books, 1985.

[15] S. Kattwinkel. *Audience participation: Essays on inclusion in performance*. Number 101. Greenwood Publishing Group, 2003.

[16] S. Koelsch. Brain correlates of music-evoked emotions. *Nat Rev Neurosci*, 15(3):170–180, 03 2014.

[17] J.-P. Lambert, S. Robaszkiewicz, and N. Schnell. Synchronisation for distributed audio rendering over heterogeneous devices, in html5. In *Proc. of Web Audio Conference (WAC)*, Atlanta, USA, 2016.

[18] S. Lee and G. Essl. Web-based temporal typography for musical expression and performance. In E. Berdahl and J. Allison, editors, *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 65–69, Baton Rouge, Louisiana, USA, May 2015. Louisiana State University.

[19] S. W. Lee, A. D. de Carvalho Jr, and G. Essl. Crowd in c [loud]: Audience participation music with online dating metaphor using cloud service. In *Web Audio Conference*, Atlanta, Georgia, USA, 2016. Georgia Institute of Technology.

[20] L. A. Liikkanen, H. Kilpiö, L. Svan, and M. Hiltunen. Lean UX: The Next Generation of User-centered Agile Development? In *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational*, NordiCHI '14, pages 1095–1100, New York, NY, USA, 2014. ACM.

[21] M. McLuhan. *Understanding media: The extensions of man*. MIT press, 1994.

[22] J. Plaza. *Tradução intersemiótica*, volume 93. Editora Perspectiva, 1987.

[23] J. Radbourne, K. Johanson, H. Glow, and T. White. The audience experience: measuring quality in the performing arts. *International Journal of Arts Management*, 11(3):16–29, 2009.

[24] C. Roberts and J. Kuchera-Morin. Gibber: Live coding audio in the browser. In *Proceedings of the International Computer Music Conference (ICMC)*, Ljubljana, Slovenia, 2012. IRZU - the Institute for Sonic Arts Research, International Computer Music Association.

[25] C. Roberts, G. Wakefield, and M. Wright. The web browser as synthesizer and interface. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 313–318, Daejeon, Republic of Korea, May 2013. Graduate School of Culture Technology, KAIST.

[26] E. Ruder. *Typography: A Manual of Design: A Textbook of Design*. Verlag Niggli, Niederteufen, 4th edition edition, Jan. 2009.

[27] T. Turino. *Music as Social Life: The Politics Of Participation*. University of Chicago Press, 2008.

[28] N. Weitzner, J. Freeman, S. Garrett, and Y.-L. Chen. massmobile -an audience participation framework. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, Ann Arbor, Michigan, 2012. University of Michigan.

[29] N. Weitzner, J. Freeman, S. Garrett, and Y.-L. Chen. massmobile-an audience participation framework. In *Proceedings of the international conference on New Interfaces for Musical Expression*, pages 21–23, 2012.

[30] Y. Wu, L. Zhang, N. Bryan-Kinns, and M. Barthet. Open symphony: Creative participation for audiences of live music performances. *IEEE Multimedia Magazine*, (48-62), 2017.

[31] L. Wyse and S. Subramanian. The viability of the web browser as a computer music platform. *Computer Music Journal*, 37(4):10–23, 2013.

[32] L. Zhang, Y. Wu, and M. Barthet. A web application for audience participation in live music performance: The Open Symphony use case. In *Proc. of International Conference on New Interfaces for Musical Expression (NIME)*, 2016.

F.13 Audio Mostly 2017 - Open Band: A Platform for Collective Sound Dialogues

Paper details

Title: *Open band: A Platform for Collective Sound Dialogues*
Authors: Ariane Stolfi, Fábio Goródscy, Antonio Deusany de Carvalho Junior, Mathieu Barthet

Conference details

Title: Audio Mostly
Venue: Centre for Digital Music, Queen Mary University of London, London, UK
Dates: August 23 to 26, 2017

Open Band: A Platform for Collective Sound Dialogues

Ariane Stolfi[*]
University of São Paulo
São Paulo, Brazil
arianestolfi@gmail.com

Mathieu Barthet
Centre for Digital Music
Queen Mary University of London
London, UK
m.barthet@qmul.ac.uk

Fábio Goródscy[†]
Instituto de Matemática e Estatística
University of São Paulo
São Paulo, Brazil
fabiog@ime.usp.br

Antonio Deusany de Carvalho Junior
Instituto de Matemática e Estatística
University of São Paulo
São Paulo, Brazil
dj@ime.usp.br

ABSTRACT

Open Band is a web-based platform for collective "sound dialogues" designed to provide audiences with empowering experiences through music. The system draws on interactive participatory art and networked music performance by engaging participants in a sonic web "agora" in collocated and/or remote gatherings, regardless of musical level. In this paper, we present our artistic intent grounded in Eco's concept of Open Works and the initial design of a web-based open environment that supports social musical interactions. Interaction operates by means of a multi-user live chat system that renders textual messages into sounds. Feedback gathered across several public participatory performances was overall positive and we identified further design challenges around personalization, crowd dynamic and rhythmic aspects.

CCS CONCEPTS

• **Human-centered computing** → *Mobile devices*; Interaction design theory, concepts and paradigms; • **Applied computing** → **Performing arts**; **Media arts**; **Sound and music computing**; Language translation;

KEYWORDS

audience participation, interactive music, web audio, performance, participatory experiences

ACM Reference Format:
Ariane Stolfi, Mathieu Barthet, Fábio Goródscy, and Antonio Deusany de Carvalho Junior. 2017. Open Band: A Platform for Collective Sound Dialogues. In *Proceedings of AM '17, London, United Kingdom, August 23–26, 2017,* 8 pages.
https://doi.org/10.1145/3123514.3123526

1 INTRODUCTION

Amongst the factors that can keep people away from playing music, are to be noted the poor access to musical instruments due to cost and the time and complexity required to learn an instrument. A growing number of works point to web audio technologies as a basis for developing more accessible interfaces for sonic or musical expression [24], since they are ubiquitous on modern computers and mobile devices and easy to distribute [17]. Furthermore, such technologies can be used to design interactive systems, independent of any extra software installation [30].

This work is part of a larger project that aims to use interactive web audio technologies to build an "open" artistic work following some of the principles advanced by Umberto Eco [7], an art work in motion, that can be shaped by each audience participant who interacts with it. We apply these principles to design a web-based multi-user musical instrument for experimental collective music improvisations. The system facilitates participatory performances which are opened to any willing person regardless of musical level or background, yet requiring devices with access to a web browser and internal loudspeakers (such as smartphones,

[*]Also composer of the pieces.
[†]Main programmer of the software.

AM '17, August 23–26, 2017, London, United Kingdom

ACM ISBN 978-1-4503-5373-1/17/08…$15.00
https://doi.org/10.1145/3123514.3123526

tablets or laptops). Musical interaction is operated through the production of text messages, a technique users of modern communication devices are usually familiar with.

The remainder of this paper is organized as follows. In Section 2, we present related works on digital participatory art and networked music performance. Section 3 describes the design of our initial prototype from the artistic intent to the technological requirements. The web-based client/server architecture of the system is detailed in Section 4. In Section 5, we describe a use case and associated performances making use of pre-established sample packs. We provide our conclusions and outline the future work we envision in Section 6.

2 RELATED WORKS

Participatory Art and Eco's Open Works

This work inscribes itself within participatory art and Eco's vision of Open Works which we introduce in this section. Western performing arts (music, dance, theatre, etc.) practices have traditionally restricted creative interventions from audiences. Contemporary Western performances are predominantly presentational, with performers preparing and providing music for another group, the audience [26].

In contrast to presentational performances, which intrinsically create a divide between audience and performers, participatory performances [14] seek to blur the boundaries between audiences and performers by giving equal role to participants, building up on the notion of "communitas" (unstructured communities in which all members are equal allowing them to share a common experience). Such aspiration from participatory art corroborates Eco's conceptions of "open works", which, according to Robey in [5], require of the public *"a much greater degree of collaboration and personal involvement than was ever required by the traditional art of the past"*.

In Open Works, it is *"the artist's decision to leave the arrangement of some of their constituents either to the public or to chance, thus giving them not a single definitive order but a multiplicity of possible orders"*. These motivations interestingly resonate with the contemporary perception of audience's role changing from primarily passive to one "co-creating values" [23], with audiences who increasingly want to "shape" their own experiences. Another important rationale of participatory art is to improve audience cognitive engagement in performance through an active form spectatorship involving a physical engagement in the performance [14].

Participatory Music Performance Using Web Technologies

Human-computer interaction (HCI) and communication technologies provide great potential to facilitate participatory art forms in our digital age as they can be used to mediate and transform creative information remotely, in (quasi) real-time and scalable ways. Within the field of digital participatory art, several authors investigated how to create participatory music performances using web technologies (see [29] for a review of some of them). Web applications for participatory performances have been proposed e.g. in [3, 8, 16, 28, 31], each of these studies manifesting specific artistic and audience creative agency models.

The massMobile framework [28] was used to let audiences control stage lighting effects or create musical loops from instructions provided through a graphical user interface (GUI) on their smartphones. Mood Conductor [8] also relies on the use of smartphones to enable audience members to conduct performers in terms of emotional directions. In Open Symphony [29, 31], audience members can generate live graphic scores interpreted by performers by voting for music playing modes on their smartphones. In this work and others, such as A.bel [3] and the CoSiMa framework [16], smartphones are employed both as a sound control and diffusion interface leveraging the loudspeakers embedded in the devices.

3 DESIGN OF THE OPEN BAND SYSTEM

Artistic Intent

A Web "Agora". This project grew out from the discernment of an implicit contradiction with contemporary social media. Although social media provide ways for people to be highly connected, they may also induce a form of isolation, inciting to be frequently absorbed in one's own individual device. In this work, we are attempting to build a collective experience that could break that isolation and put every members of an audience in a web "agora".

Eco's Framework. We are framing our work within Eco's conception of "open work" [6]. Being "in movement", open works can be changed by every participant, providing freedom to the performer to change it or to the audience to interpret. In Open Band, we are inviting the audience to join as performers, since everything typed in gets converted into sounds that are played back to everyone. Eco's vision of open work includes systems with a certain degree of indeterminacy, ambiguity and even deliberate "dis-ordering". Such principles occur in Open Band, especially when a large number of users interact. The project involves an anonymous multi-user chat, where everyone can be free to express themselves without risk of being pointed at or judged.

Mapping Language to Sounds. As pointed by Koelsch [15], music is a social activity that puts people into contact with one another, and that can lead to "increased social cohesion of a group" and promote cooperation between individuals. As the author says, making music collectively generally involves

a high degree of coordination of actions, e.g. to synchronize to musical beats. In this project, we are proposing a system where participants don't need to have previous musical skills. The act of writing, seen as a "technology" in McLuhan and McLuhan [19], can be used as a way to crystallize verbal sounds into the visual space. Unlike speech recognition systems that convert speech into text, we are atomizing texts to trigger sounds one after the other, to generate music. Taken as separated letters, words are played as musical phrases, which thus give non musicians access to simple musical experiences. When participants start to talk one after the other using the chat, messages get superimposed. The rhythm becomes mores frenetic and it becomes more difficult to recognize how specific messages are sounding, like when several people talk at the same time.

Participatory Music Performance. Being hosted in a web chat, the system can be easily accessible to anyone having access to a device with a web browser (e.g. computer, laptop, tablet or smartphone) compatible with web audio technologies, and an Internet connection. Once in the chat, everything that is typed in can be heard by anyone connected to the same message server. For participatory performance, we deploy a version of the server on a local host, to create a closed network of people and to lower the bandwidth requirements for the audience and reduce delay time in playing the messages. We usually connect one of the devices to a PA sound system to increase the volume and thus the musical perception. A performance can have a conductor who uses special commands to change the samples and to change the overall dynamics of the piece. The conductor can also completely turn off the sound which may be used at times. There is no hierarchy amongst audience participants, and everything which is typed in is played with no distinction.

In another use case, the system can act as an instrument, in solo performances or within free improvisation contexts, for musicians who know the samples and the commands to change sounds. Currently, the system is closed with some musical choices previously defined by composers, that we will discuss in Section 5, but the samples are easily changeable by replacing files in folders on the server side.

Technological Design

To ease audience participation, the project is presented as a website, with no login and with open access. During performances, a person also acting as conductor typically runs the site on a local host, and provides a private WiFi network for the audience. The website can be projected on a screen using a projector to provide visual feedback and a PA can be used to provide aural feedback, which also allows to share the performance to those not joining the chat.

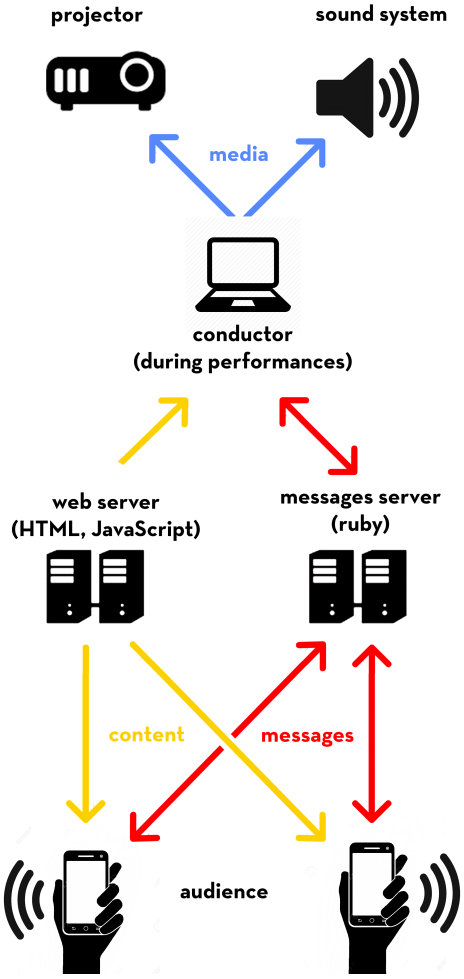


Figure 1: Overview of the Open Band system. The audience interacts with the system trough a website. During performances, a participant can act as conductor to change dynamics and sound packs. The website can be projected on a screen and sounds diffused using a PA.

Since text messages are probably the main form of communication on smartphones nowadays, we chose to experiment with this type of communication as input to produce interactive music. Many kinds of interfaces use typing as input,

such as those used in live coding [4], in audience participation pieces to receive feedback from the audience [21], to get user data as source for algorithmic compositions, or using keyboard as music controllers [9].

We addressed these requirements by developing a client/server web-based architecture which is described in the next section. The project was developed following an Agile methodology called Lean Ux [18], that consists in having always a working version of the project since its start, and to perform gradual changes for each performance as we receive feedback of the audience and the composer.

4 OPEN BAND SYSTEM WEB ARCHITECTURE

The project was built having two server-side components that are independent: one of them is a typical web server that serves web content such as HTML, JavaScript and audio files and is based on the Web Audio API [27]; the other server component is responsible for making messages deliverable (see Figure 1). This is a desirable feature, just as it makes possible to change the messaging technology at any moment, as well as allows different front-ends to receive or send messages.

Front-end

The front end was designed to be as minimalist as possible, with only a text input for the messages. The idea was to use minimal design elements to avoid distraction from the main content of the messages sent and the sounds generated from them in real time. There is no register nor login windows, and the site is totally open for every user.

Interface and Interaction. The interaction process happens through an open chat interface, where there is no distinction between users. Hence it is not possible to identify who is playing what. Each message sent to the chat is loaded into a chat room and played as a sequence of samples with no hierarchy. In an inter-semiotic translation [22] between text and sound, we decode the text messages as musical information, through concatenation of samples. Messages can be played again by touching or clicking them on screen. Unlike speech synthesis mechanisms, our sound chat does not concatenate syllables, so words cannot be understood as words, but only as sound rhythmic sequences. Besides the sounds generated from the textual inputs of participants, there are also commands that can be live coded. Those commands are hidden from the public to create some degree of surprise when used by a participant acting as conductor. Participant-conductors can change the audio samples pack, change the global volume or turn the sound off.

Open Source Project. A second project is maintained [12] as an alternative for people who would like to reuse the project without having to amend the server component. It is possible to change the audio files and play one's own performances.

The current implementation treat all provided sounds as samples, and each ASCII character is associated to a sample (mp3 audio files used by the system are named after the given ASCII number they relate to). When a user enters the chat, the audio samples are loaded into a buffer, and they are played when messages are sent. In the version used in performances so far, samples are played one after another, as indicated by the sequence of the letters on the messages.

Back-end

The current implementation of the messaging server is written in Ruby, making use of the Puma, Sinatra and Faye frameworks, as well as the WebSocket technology for server/browser communication. The code has been developed with the aim of reproducibility on Linux platforms and is released as open source [13]. The messaging technology plays a main role in such application and early versions of the project frameworks like PubNub and Peer.js have been tested. However the Ruby server was eventually kept to enable performances even without any Internet access, what could not be achieved with the other frameworks.

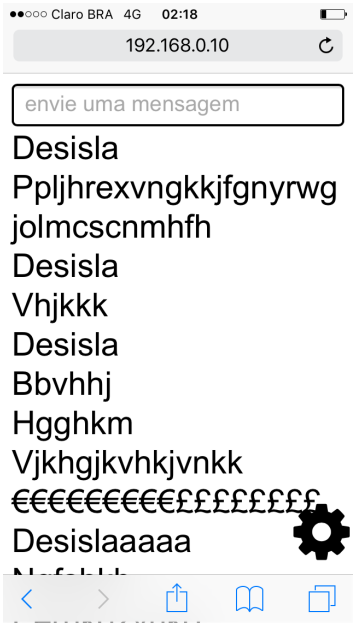


Figure 2: Interface of the Open Band chat on mobile device.

5 PARTICIPATORY PERFORMANCE USE CASES

Sample Design

In the version of the project presented in this paper, all the sounds are played as samples, and each sample pack works as an interactive music composition by itself. As the order and nature of the sounds played by the audience is not known in advance, we cut and produced samples so that they could either be played alone or combined with one another. To this end, we cut the samples based on James Tenney's concept of clang, or aural gestalts, a minimum unity of a sound object, that is similar to Pierre Schaeffer's concept of "cellule" sound-objects, short or redundant objects [20]. Being simple sound units, we believe that such sounds can be more suited to random combinations than complex sounds, with less risk of rhythmic or tonal inconsistencies.

We are also making an analogy with the phonetic alphabet "technology" itself based on the idea of atomization of the spoken language [19]. To leave room for chance, we tried to cover high degree of variation of sound clangs, and made four different compositions of packs of samples, with different sonorities, that can be changed in real time by typing in special commands during the chat (see the description of "samples3/" below). To maintain a certain rhythm, the samples were cut in fixed and multiple formats, in order to enable them to fit within a time grid.

Galaxies (samples0/). The default pack was sought to maintain some phonetic relation with original letters, to introduce public to the system. Most of "lowercase letter sounds" were cut from Haroldo de Campos readings of his long poem Galáxias [2]. This was achieved to also pay tribute to concrete poetry defined in Campos et al. [1] as a "tension of thing-words in time space. Dynamic structure: multiplicity of concomitant movements". For special characters, we implemented in this pack symbolic semantic-sound relationships such as the use of a cash machine sound for the dollar sign. We also use other more iconic analogies, such as using subtractive graphic synthesis made by spectral visuals, as illustrated in Figure 3. This sample pack is the default one and is generally used for introductory parts in the performances to make the audience familiar with the sound-to-letter associations on the website and during the performance.

Percussion and Chords (samples1/). The second pack of samples deals more with materials in the realm of pop electronic music, such as elements of traditional electronic samplers, chords and guitar samples. We associated percussion material to consonants, trying to identify sounds that resemble the original phonetic of the letters, such as bass kicks for the "B" and cymbal for the "S". For vowels, we used synthesized sound made from additive samples in a C major scale. For some special characters, we took samples from Velimir Khlebnikov's Radio of The Future [25].

Collaborative Samples Set (samples2/). This sample set was put together by Viktor Kisil, professor of Music Production at Anhembi Morumbi University (São Paulo, Brazil), who asked his students to make some short samples for a performance. This pack results from a collaboration in mapping the samples with the ASCII table. It contains samples from different kinds of sources, concrete and synthetic sounds, and also includes a lot of special sound effects. Relations between sounds and letters are more arbitrary than in the other sample packs.

Orquestra Errante (samples3/). These samples were cut from recordings of free improvisation group Orquestra Errante, conducted by Rogerio Costa at University of São Paulo, mostly produced on traditional acoustic instruments, such as saxophone, flute, clarinet, trombone, guitar, percussion and voice. The instruments were often played with extended performance techniques during rehearsals or in studio. We searched an archive of several past free improvisations of the ensemble, to find sounds evoking letters, or repetitions

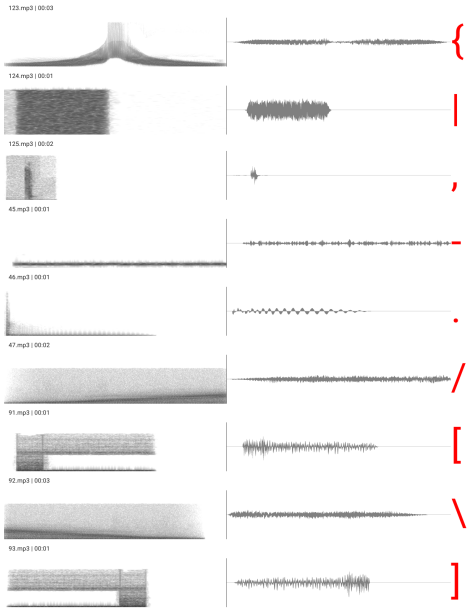


Figure 3: Spectrographs and waveshapes of some of the samples from the first sample pack. These were obtained by subtractive graphic synthesis.

for numbers or that evoked the shape of the characters for some special ones.

Overall Performance Apparatus

The performances are designed to be made in any space that can fit a laptop and a network hub. All interested participant are invited to join the network with their device and connect to the the application in a web browser. As web browsers are now ubiquitous, the only limitation for connecting to the application is the device being able to join the network through WiFi. A regular laptop should be able to run the application without any issues. The last performances were made on a laptop with AMD A4-5000 processor and 4GB memory. Any devices that can install Ruby gems should be able to host the server. The source code is hosted on Github as open source software and there is a shell script with the application to easily start the server. In most performances held so far, we used a projection of the chat on a screen.

Performance Procedure

Audiences were first asked to join a specific Wifi network running the server, and then enter an IP address that was provided to them (e.g. in the projection). As users enter the web page, they start to write messages trough the chat. At some point, a participant acting as conductor (one of the facilitators can hold this role) change the samples to cause some degree of surprise. The performances usually last for about 20 minutes and finish when the conductor types the command "stop!", witch brings the system to silence.

Settings and Participants

To date, Open Band has been performed at five occasions, namely at the Bigorna Festival, at a public square (see Figure 4), at various concerts and in private spaces. During the first performance at the Bigorna Festival (see Figure 4), the audience included about 70 people, 30 of whom were able to connect to the chat (due to restrictions of our current router we could not support more connections). The audience was formed mostly by students of Music Production that had collaborated on the third sample packs.

During the second performance at the ABRAPEN conference, the audience mostly comprised traditional musicians who were conference attendees. There was about 25 participants in the audience who demonstrated their interest by asking information about the system and about the sounds chosen through the chat. There was more interest by the audience in trying sound combinations together than by just exchanging textual messages anonymously [1].

[1] A recording of the performance at ABRAPEN is available at: https://www.youtube.com/watch?v=NOWapLq6eiU

Figure 4: People interacting during an Open Band performance at a public square.

The third performance was held on a web radio. 10 participants actively joined the chat and 20 participants listened to the performance through the radio [2].

The fourth performance was held at a concert of experimental music in a bar, where there was about 40 participants [3]. The audience was formed by experimental musicians and enthusiasts, who kept experiencing with sound combinations, more than with discourse.

The last performance held to date was conducted as part of a Computer Music Concert at the *Instituto de Matemática e Estatística* (IME), with an audience composed mostly of programmers and computer scientists[4]. During this performance, some of the participants discovered a flaw on our site's security that made possible to send JavaScript and CSS

[2] A recording of the radio session can be heard at: https://soundcloud.com/asss/banda-aberta-na-radiograve-reset-do-universo

[3] A video of the screen recording of the performance at the bar is available at: https://www.youtube.com/watch?v=DpCuU41tWM8

[4] A video of the screen at IME's performance is available at: https://www.youtube.com/watch?v=xs23z1IfPfY

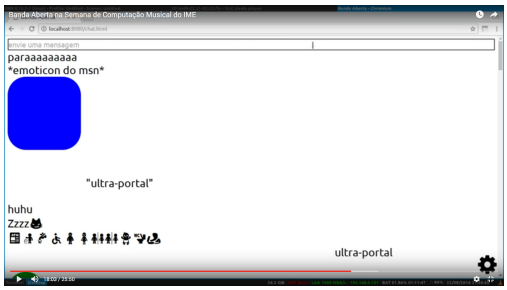


Figure 5: Screenshot during the performance at IME, with people inserting CSS data on the chat.

commands through the chat interface, as shown in Figure 5. Participants kept trying to make things move on the screen and change visual aspects of the site, despite the sounds that were been played. This illustrated a surprising example of use of our system with audiences using their creative agency in a collaborative way.

Qualitative Evaluation

We discuss here several aspects that emerged across the five performances presented in the previous section. The system was used in different ways depending on the profile of the audiences. We could see that in musical contexts, the audiences spend more time experiencing with the samples and rhythm, typing phrases without meaning. However with a younger public, the participants tended more to play by discussing in anonymous ways. As conversations got more intense, layers of sounds overlapped, creating more frenetic rhythms.

Noticeably, in all instances, we received a lot of feedback from the public through the chat, showing interest from the participant who supported the project. The engagement of active participants was generally high, but some users complained that it was difficult at times to identify which sounds corresponded to which message, diminishing their sense of creative agency. Specifically when there were more people that our router could support, like during the Bigorna Festival, there was some frustration from users who could not join the chat.

The anonymity of the chat is also a disputable point, as sometimes the users used the platform to make political discourse and wanted to have their messages associated to them. At the Festival Bigorna where the audience was very young, there was unfortunately some case of homophobic and misogynist discourse employed by the participants. Our current systems does not have a straightforward way to prevent such messages to be sent. However, besides such issues, we believe that the anonymity benefits a high engagement, since people do not get ashamed of saying things and therefore producing sounds through the chat.

After the concert at IME, five people from the audience completed an online survey about the project. No user found it difficult to use the application, two of them felt musically limited by the resources of the system, and everyone felt that they were collaborating in the process during the performance.

6 CONCLUSIONS AND FUTURE WORK

We presented in this paper Open Band, a novel system to create participatory music performances by sonifying participants' textual messages in a live chat platform. The free interaction and ease of use has shown to provide a high degree of fun for some participants during five performances which were held in various settings. The system fosters a sense of play that resembles that of games, but also affords socio-political standpoints as expressed by free speech in a chat system. Messages decoded into music also give the possibility of communicating beyond symbolic written language sharing of phrases purely at the sonic level, without expected literal sense.

To date, Open Band has been developed mainly as a performance project and creative coding experiment, with fixed collections of samples. We seek in the future to expand it to be a framework to be used in different musical contexts. Also it is quite simple for any enthusiast to change the sample packs in the open-source software.

In future work, we wish to use the Audio Commons API[5] and/or the Freesound API [11] that grants access to a collection of more than 300k sound samples released under Creative Commons licenses [10], enabling interesting media re-purposing. We are also interested in developing more complex text-to-audio mappings, to use gestural controls from users, and real-time analysis of the overall dynamics in the text flow produced by the crowd (mapping textual dynamics to musical dynamics).

ACKNOWLEDGMENTS

We acknowledge support from the NuSom Research group at University of São Paulo and the CAPES grant awarded to Ariane Stolfi. Mathieu Barthet acknowledges support from the EU H2020 Audio Commons grant (688382).

REFERENCES

[1] Augusto de Campos, Haroldo de Campos, and Decio Pignatari. 2014. *Teoria Da Poesia Concreta. Textos Criticos E Manifestos 1950-1960* (literatura brasileira edition ed.). Atelie.

[2] Haroldo de Campos. 2004. *Galáxias: Inclui o CD Isto Não é um Livro de Viagem* (1 edition ed.). Editora 34, São Paulo, SP.

[3] A. Clément, F. Ribeiro, R. Rodrigues, and R. Penha. 2016. Bridging the gap between performers and the audience using networked smartphones: the a.bel system. In *Proceedings of International Conference on Live Interfaces.*

[4] Nick Collins, Alex. McLean, Julian. Rohrhuber, and Adrian. Ward. 2003. Live coding in laptop performance. *Organised Sound* 8, 03 (2003), 321–330.

[5] Umberto Eco. 1989. *The Open Work (Translated by Anna Cancogni With an Introduction by David Robey).* Harvard University Press, Cambridge, Massachusetts, USA.

[6] Umberto Eco. 2006. The Poetics of the Open Work. 1962. *Participation: Documents of Contemporary Art* (2006), 20–40.

[7] Umberto Eco. 2015. *Obra Aberta.* Perspectiva.

[8] G. Fazekas, M. Barthet, and Mark B. Sandler. 2013. The Mood Conductor System: Audience and Performer Interaction using Mobile Technology and Emotion Cues. In *Proc. of the Int. Symposium on Computer Music Multidisciplinary Research (CMMR).* http://www.cmmr2013.cnrs-mrs.fr/Docs/CMMR2013Proceedings.pdf

[5] http://www.audiocommons.org

[9] Rebecca Fiebrink, Ge Wang, and Perry R. Cook. 2007. Don't Forget the Laptop: Using Native Input Capabilities for Expressive Musical Control. In *Proceedings of the 7th International Conference on New Interfaces for Musical Expression (NIME '07)*. ACM, New York, NY, USA, 164–167. https://doi.org/10.1145/1279740.1279771

[10] Frederic Font, Gerard Roma, and Xavier Serra. 2013. Freesound Technical Demo. In *Proceedings of the 21st ACM International Conference on Multimedia (MM '13)*. ACM, New York, NY, USA, 411–412. https://doi.org/10.1145/2502081.2502245

[11] Freesound. 2014. APIv2 Overview - Freesound API documentation. (2014). https://www.freesound.org/docs/api/overview.html

[12] Fabio Gorodcy. 2016. fabiogoro/banda. (2016). https://github.com/fabiogoro/banda

[13] Fabio Gorodscy. 2016. fabiogoro/bandaserver. (2016). https://github.com/fabiogoro/bandaserver

[14] Susan Kattwinkel. 2003. *Audience participation: Essays on inclusion in performance*. Number 101. Greenwood Publishing Group.

[15] Stefan Koelsch. 2014. Brain correlates of music-evoked emotions. *Nat Rev Neurosci* 15, 3 (03 2014), 170–180. http://dx.doi.org/10.1038/nrn3666

[16] J.-P. Lambert, S. Robaszkiewicz, and N. Schnell. 2016. Synchronisation for Distributed Audio Rendering over Heterogeneous Devices, in HTML5. In *Proc. of Web Audio Conference (WAC)*. Atlanta, USA.

[17] Sang Won Lee and Georg Essl. 2015. Web-Based Temporal Typography for Musical Expression and Performance. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME 2015)*. The School of Music and the Center for Computation and Technology (CCT), Louisiana State University, Baton Rouge, Louisiana, USA, 65–69. http://dl.acm.org/citation.cfm?id=2993778.2993798

[18] Lassi A. Liikkanen, Harri Kilpiö, Lauri Svan, and Miko Hiltunen. 2014. Lean UX: The Next Generation of User-centered Agile Development?. In *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational (NordiCHI '14)*. ACM, New York, NY, USA, 1095–1100. https://doi.org/10.1145/2639189.2670285

[19] Eric McLuhan and Marshall McLuhan. 1992. *Laws of Media: The New Science* (edição: 2nd revised ed. ed.). University of Toronto Press, Scholarly Publishing Division, Toronto.

[20] Michel Chion. 2009. *Guide to Sound Objects: Pierre Schaeffer and Musica Research*. London.

[21] Female Laptop Orchestra. 2016. In Transglasphone. CMMR. (2016). http://www.crisap.org/event/in-transglasphone/

[22] Julio Plaza. 1987. *Tradução intersemiótica*. Vol. 93. Editora Perspectiva.

[23] Jennifer Radbourne, Katya Johanson, Hilary Glow, and Tabitha White. 2009. The audience experience: measuring quality in the performing arts. *International Journal of Arts Management* 11, 3 (2009), 16–29. http://www.jstor.org/stable/41064995

[24] Charles Roberts, Graham Wakefield, and Matthew Wright. 2013. The Web Browser As Synthesizer And Interface. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Graduate School of Culture Technology, KAIST, Daejeon, Republic of Korea, 313–318. http://nime.org/proceedings/2013/nime2013_282.pdf

[25] UbuWeb Sound. 2017. Sound Experiments in The Russian Avant-Garde (1908-1942). (2017). http://www.ubu.com/sound/russian_avant.html

[26] T. Turino. 2008. *Music as Social Life: The Politics Of Participation*. University of Chicago Press.

[27] W3C. 2014. Web Audio API. (2014). https://webaudio.github.io/web-audio-api/

[28] Nathan Weitzner, Jason Freeman, Stephen Garrett, and Yan-Ling Chen. 2012. massMobile – an Audience Participation Framework. In *Proc. of the International Conference on New Interfaces for Musical Expression*. 92–95.

[29] Y. Wu, L. Zhang, N. Bryan-Kinns, and M. Barthet. 2017. Open Symphony: Creative Participation for Audiences of Live Music Performances. *IEEE Multimedia Magazine* 48-62 (2017). http://ieeexplore.ieee.org/document/7849101/

[30] Lonce Wyse and Srikumar Subramanian. 2013. The viability of the web browser as a computer music platform. *Computer Music Journal* 37, 4 (2013), 10–23.

[31] Leshao Zhang, Yongmeng Wu, and Mathieu Barthet. 2016. A Web Application for Audience Participation in Live Music Performance: The Open Symphony Use Case. In *Proc. of International Conference on New Interfaces for Musical Expression (NIME)*.

Bibliography

Akkermann(2014) Miriam Akkermann. Computer network music approximation to a far-scattered history. Em *Proceedings of the EMS14 - Electroacoustic Music Beyond Concert Performance*, pages 1–7. Electroacoustic Music Studies Network. Cited at pag. 7

Alliance(2016) WiFi Alliance. Wi-fi peer-to-peer (p2p) technical specification v1. 7. *Wi-Fi Alliance Specification*, 1:1–201. Cited at pag. 21

Allison and Dell(2012) Jesse Allison and Christian Dell. Aural: A mobile interactive system for geo-locative audio synthesis. Em *Proceedings of the International Conference on New Interfaces for Musical Expression*, Ann Arbor, Michigan. University of Michigan. URL http://www.nime.org/proceedings/2012/nime2012_301.pdf. Visited on May, 2017. Cited at pag. 13

Allison *et al.***(2013)** Jesse Allison, Yemin Oh and Benjamin Taylor. Nexus: Collaborative performance for the masses, handling instrument interface distribution through the web. Em *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 1–6, Daejeon, Republic of Korea. Graduate School of Culture Technology, KAIST. URL http://nime.org/proceedings/2013/nime2013_287.pdf. Visited on May, 2017. Cited at pag. 10, 14, 15

Ananya *et al.***(2008)** Misra Ananya, Georg Essl and Michael Rohs. Microphone as sensor in mobile phone performance. Em *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 185–188, Genoa, Italy. URL http://www.nime.org/proceedings/2008/nime2008_185.pdf. Visited on May, 2017. Cited at pag. 13

Arango *et al.***(2013)** Julian Jaramillo Arango, Marcio Tomiyoshi, Fernando Iazzetta and Marcelo Queiroz. Brazilian Challenges on Network Music. Em *Sound and Music Computing Conference*, pages 453–459. Cited at pag. 9

Arango(2014) Julián Jaramillo Arango. *NETWORK MUSIC: criação e performance musical colaborativa no âmbito das redes de informação*. PhD Thesis, Escola de Comunicação e Artes, Universidade de São Paulo, Brasil. Cited at pag. 9

Balasubramanian *et al.***(2009)** Niranjan Balasubramanian, Aruna Balasubramanian and Arun Venkataramani. Energy consumption in mobile phones: A measurement study and implications for network applications. Em *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, IMC '09, pages 280–293, New York, NY, USA. ACM. ISBN 978-1-60558-771-4. URL http://doi.acm.org/10.1145/1644893.1644927. Visited on March, 2017. Cited at pag. 22

Baldan *et al.***(2013)** Stefano Baldan, Stefania Serafin and Amalia De Götzen. Melody bounce: Mobile rhythmic interaction for children. Em *Proceedings of the International Conference on Sound and Music Computing (SMC)*, pages 197–200. Royal Institute of Technology (KTH), and the Department of Composition, Conducting and Music Theory at the Royal College of Music (KMH). Cited at pag. 6

Bandeira and de Carvalho Junior(2014) André Damião Bandeira and Antonio Deusany de Carvalho Junior. Notes on the elimination of the mobile music audience. Em *The Fourteenth Biennial*

Symposium on Arts and Technology, Connecticut College, New London, CT, USA. Ammerman Center for Arts and Technology. Cited at pag. 11, 55, 80

Barbosa(2003) Alvaro Barbosa. Displaced soundscapes: A survey of network systems for music and sonic art creation. *Leonardo Music Journal*, 13:53–59. Cited at pag. 9

Barroso and Hölzle(2007) Luiz André Barroso and Urs Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37. ISSN 0018-9162. Cited at pag. 10

Bartlette *et al.***(2006)** Christopher Bartlette, Dave Headlam, Mark Bocko and Gordana Velikic. Effect of network latency on interactive musical performance. *Music Perception: An Interdisciplinary Journal*, 24(1):49–62. ISSN 07307829. Cited at pag. 22

Bhagwat(2001) P. Bhagwat. Bluetooth: technology for short-range wireless apps. *IEEE Internet Computing*, 5(3):96–103. ISSN 1089-7801. Cited at pag. 20

Bianchi(2014) André Jucovsky Bianchi. *Processamento de áudio em tempo real em dispositivos computacionais de alta disponibilidade e baixo custo.* Master thesis, Universidade de Sao Paulo. Cited at pag. 10, 34, 41

Bianchi and Queiroz(2012) André Jukovsky Bianchi and Marcelo Queiroz. On the performance of real-time DSP on Android devices. Em *9th Sound and Music Computing Conference*, pages 113–120. Cited at pag. 10, 27, 74

Bluetooth(2009) SIG Bluetooth. Bluetooth core specification version 3.0 + hs. *Specification of the Bluetooth System.* Cited at pag. 20

Bluetooth(2016) SIG Bluetooth. Bluetooth core specification version 5.0. *Specification of the Bluetooth System.* Cited at pag. 20

Brinkmann(2012) Peter Brinkmann. *Making musical apps.* O'Reilly Media, Inc. Cited at pag. 12, 71

Brinkmann *et al.***(2011)** Peter Brinkmann, Peter Kirn, Richard Lawler, Chris McCormick, Martin Roth and Hans-Christoph Steiner. Embedding pure data with libpd. Em *Proceedings of the Pure Data Convention*, volume 291. Cited at pag. 10, 12

Bryan *et al.***(2010)** Nicholas J. Bryan, Jorge Herrera, Jieun Oh and Ge Wang. Momu : A mobile music toolkit. Em *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 174–177, Sydney, Australia. URL http://www.nime.org/proceedings/2010/nime2010_174.pdf. Visited on March, 2017. Cited at pag. 13

Carôt and Werner(2008) Alexander Carôt and Christian Werner. Distributed network music workshop with soundjack. *Proceedings of the 25th Tonmeistertagung, Leipzig, Germany.* Cited at pag. 9

Carôt(2010) Alexander Carôt. Low latency audio streaming for internet-based musical interaction. *Streaming Media Architectures, Techniques, and Applications: Recent Advances: Recent Advances*, page 362. Cited at pag. 9

Carôt and Werner(2007) Alexander Carôt and Christian Werner. Network music performance-problems, approaches and perspectives. Em *Proceedings of the "Music in the Global Village" - Conference*, volume 162, pages 23–10, Budapest, Hungary. Cited at pag. 9

Carôt and Werner(2009) Alexander Carôt and Christian Werner. Fundamentals and principles of musical telepresence. *Journal of Science and Technology of the Arts*, 1(1):26–37. Cited at pag. 9

Carôt *et al.*(2007) Alexander Carôt, Pedro Rebelo and Alain Renaud. Networked music performance: State of the art. Em *Audio engineering society conference: 30th international conference: intelligent audio environments*. Audio Engineering Society. Cited at pag. 9

Chafe and Gurevich(2004) Chris Chafe and Michael Gurevich. Network time delay and ensemble accuracy: Effects of latency, asymmetry. Em *Audio Engineering Society Convention 117*. URL http://www.aes.org/e-lib/browse.cfm?elib=12865. Visited on May, 2017. Cited at pag. 22

Choi and Berger(2013) Hongchan Choi and Jonathan Berger. Waax: Web audio API extension. Em *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 499–502, Daejeon, Republic of Korea. Graduate School of Culture Technology, KAIST. URL http://nime.org/proceedings/2013/nime2013_119.pdf. Visited on March, 2017. Cited at pag. 11, 87

Cook and Scavone(1999) Perry R. Cook and Gary Scavone. The synthesis toolkit (stk). Em *Proceedings of the International Computer Music Conference*, pages 164–166. International Computer Music Association. Cited at pag. 10, 12

Corby(2013) Tom Corby. *Network Art: Practices and Positions*. Innovations in Art and Design. Taylor & Francis. ISBN 9781136578052. Cited at pag. 7

Coskun *et al.*(2013) Vedat Coskun, Busra Ozdenizci and Kerem Ok. A survey on near field communication (nfc) technology. *Wireless Personal Communications*, 71(3):2259–2294. ISSN 1572-834X. URL http://dx.doi.org/10.1007/s11277-012-0935-5. Visited on March, 2017. Cited at pag. 20

Cotton *et al.*(2011) M. Cotton, L. Eggert, J. Touch, M. Westerlund and S. Cheshire. Internet assigned numbers authority (IANA) procedures for the management of the service name and transport protocol port number registry. RFC 6335, RFC Editor. URL http://www.rfc-editor.org/rfc/rfc6335.txt. Visited on May, 2017. Cited at pag. 24

Coulouris *et al.*(2011) George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair. *Distributed Systems: Concepts and Design*. Addison-Wesley Publishing Company, USA, 5th edição. ISBN 9780132143011. Cited at pag. 22

CSE News(2017) CSE News. Michigan mobile phone ensemble stretches, challenges performers and audience. http://www.eecs.umich.edu/eecs/about/articles/2015/w15-mobile-phone-ensemble-performance.html, 2017. Visited on May, 2017. Cited at pag. 13

Cáceres and Chafe(2010) Juan-Pablo Cáceres and Chris Chafe. JackTrip: Under the Hood of an Engine for Network Audio. *Journal of New Music Research*, 39:183–187. Cited at pag. 22

Dahl *et al.*(2011) Luke Dahl, Jorge Herrera and Carr Wilkerson. Tweetdreams : Making music with the audience and the world using real-time twitter data. Em *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 272–275, Oslo, Norway. URL http://www.nime.org/proceedings/2011/nime2011_272.pdf. Visited on May, 2017. Cited at pag. 15

de Carvalho Junior(2013) Antonio Deusany de Carvalho Junior. Touches on the line: Sharing csound scores using a web server and mobile phones. Em *The 2nd International Csound Conference*. Berklee College of Music, Boston, Massachusetts, USA. Cited at pag. 11, 15, 54, 75

de Carvalho Junior(2014) Antonio Deusany de Carvalho Junior. Sensors2pd: Mobile sensors and wifi information as input for pure data. Em *42nd International Computer Music Conference (ICMC) joint with the 13rd Sound & Music Computing conference (SMC)*. International Computer Music Association. Cited at pag. 11, 55, 77

de Carvalho Junior(2015) Antonio Deusany de Carvalho Junior. Cooperative live coding as an instructional model. Em *Computing Conference (CLEI), 2015 Latin American*, pages 1–7. IEEE. Cited at pag. 9, 55, 82

de Carvalho Junior(2015) Antonio Deusany de Carvalho Junior. Indoor localization during installations using WiFi. Em Edgar Berdahl and Jesse Allison, editors, *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 40–41, Baton Rouge, Louisiana, USA. Louisiana State University. URL http://www.nime.org/proceedings/2015/nime2015_237.pdf. Visited on May, 2017. Cited at pag. 11, 55, 80

de Carvalho Junior and Mayer(2015) Antonio Deusany de Carvalho Junior and Thomas Mayer. Sensors2osc. Em *Sound and Music Computing Conference*. Maynooth University. Cited at pag. 15, 55, 78

de Carvalho Junior *et al.***(2013)** Antonio Deusany de Carvalho Junior, Max Rosan, André Bianchi and Marcelo Queiroz. Fft benchmark on android devices: Java versus jni. Em *XIV Simpósio Brasileiro de Computação Musical*. Sociedade Brasileira de Computação. Cited at pag. 10, 41, 54, 74

de Carvalho Junior *et al.***(2015a)** Antonio Deusany de Carvalho Junior, Sang Won Lee and Georg Essl. Supercopair: Collaborative live coding on supercollider through the cloud. Em *International Conference on Live Coding*. Cited at pag. 9, 55, 82

de Carvalho Junior *et al.***(2015b)** Antonio Deusany de Carvalho Junior, Marcelo Queiroz and Georg Essl. Computer music through the cloud: Evaluating a cloud service for collaborative computer music applications. Em *Proceedings of the International Computer Music Conference (ICMC)*, Denton, Texas, USA. University of North Texas, International Computer Music Association. Cited at pag. 33, 55

de Carvalho Junior *et al.***(2016)** Antonio Deusany de Carvalho Junior, Sang Won Lee and Georg Essl. Understanding cloud support for the audience participation concert performance of crowd in c[loud]. Em *Proceedings of the International Conference on New Interfaces for Musical Expression*, volume 16 of *2220-4806*, pages 176–181, Brisbane, Australia. Queensland Conservatorium Griffith University. ISBN 978-1-925455-13-7. URL http://www.nime.org/proceedings/2016/nime2016_paper0037.pdf. Visited on May, 2017. Cited at pag. 15, 55, 84

de Souza e Silva(2004) Adriana de Souza e Silva. Art by telephone: from static to mobile interfaces. Em Lanfranco Aceti, editor, *Leonardo Electronic Almanac*, volume 20. Leonardo Online. http://www.leoalmanac.org/leonardo-electronic-almanac-volume-12-no-10-october-2004 (Visited on May, 2017). Cited at pag. 5

Deering(1989) S.E. Deering. Host extensions for IP multicasting. RFC 1112, RFC Editor. URL http://www.rfc-editor.org/rfc/rfc1112.txt. Visited on May, 2017. Cited at pag. 24

Deicke *et al.***(2012)** F. Deicke, W. J. Fisher and M. Faulwaßer. Optical wireless communication to eco-system. Em *2012 Future Network Mobile Summit (FutureNetw)*, pages 1–8. Cited at pag. 20

Derbinsky and Essl(2012) Nate Derbinsky and Georg Essl. Exploring reinforcement learning for mobile percussive collaboration. Em *Proceedings of the International Conference on New Interfaces for Musical Expression*, Ann Arbor, Michigan. University of Michigan. URL http://www.nime.org/proceedings/2012/nime2012_241.pdf. Visited on May, 2017. Cited at pag. 13

Diot *et al.***(2000)** C. Diot, B.N. Levine, B. Lyles, H. Kassem and D. Balensiefen. Deployment issues for the ip multicast service and architecture. *Network, IEEE*, 14(1):78–88. ISSN 0890-8044. Cited at pag. 23

Duckworth(2005) William Duckworth. *Virtual Music: How the web got wired for sound.* Routledge. Cited at pag. 11

Ekeus *et al.*(**2013**) Henrik Ekeus, Samer A Abdallah, Peter W McOwan and Mark D Plumbley. How predictable do we like our music? eliciting aesthetic preferences with the melody triangle mobile app. Em *Proceedings of Sound and Music Computing Conference (SMC)*, pages 80–85. KTH Royal Institute of Technology. Cited at pag. 6

Essl and Rohs(2006) Georg Essl and Michael Rohs. Mobile STK for symbian OS. Em *Proceedings of the International Computer Music Conference (ICMC)*, pages 278–281, New Orleans, Louisiana, USA. Tulane University, International Computer Music Association. Cited at pag. 10, 12

Essl and Rohs(2007) Georg Essl and Michael Rohs. Shamus: A sensor-based integrated mobile phone instrument. Em *Proceedings of the International Computer Music Conference (ICMC)*, pages 200–203, Holmen Island, Copenhagen, Denmark. International Computer Music Association. Cited at pag. 6, 13

Feng *et al.*(**2014**) D. Feng, L. Lu, Y. Yuan-Wu, G. Y. Li, S. Li and G. Feng. Device-to-device communications in cellular networks. *IEEE Communications Magazine*, 52(4):49–55. ISSN 0163-6804. Cited at pag. 21

Fiske(2010) John Fiske. *Introduction to communication studies.* Routledge. Cited at pag. 18

Friedman *et al.*(**2013**) R. Friedman, A. Kogan and Y. Krivolapov. On power and throughput tradeoffs of wifi and bluetooth in smartphones. *IEEE Transactions on Mobile Computing*, 12(7): 1363–1376. ISSN 1536-1233. Cited at pag. 21

Gang *et al.*(**1997**) Dan Gang, Gregory V Chockler, Tal Anker, Alex Kremer and Tomas Winkler. Transmidi: a system for midi sessions over the network using transis. Em *Proceedings of the International Computer Music Conference (ICMC)*, pages 283–286, Thessaloniki, Greece. Aristotle University, International Computer Music Association. Cited at pag. 7

Gasperini(2011) Marco Gasperini. Hypothesis for the foundation of a laptop orchestra. Em *Proceedings of International Computer Music Conference (ICMC)*, pages 602–608, Huddersfield, UK. University of Huddersfield, International Computer Music Association. Cited at pag. 8

Gass and Diot(2010) Richard Gass and Christophe Diot. An experimental performance comparison of 3g and wi-fi. Em Arvind Krishnamurthy and Bernhard Plattner, editors, *Passive and Active Measurement: 11th International Conference, PAM 2010, Zurich, Switzerland, April 7-9, 2010. Proceedings*, pages 71–80, Berlin, Heidelberg. Springer Berlin Heidelberg. ISBN 978-3-642-12334-4. URL http://dx.doi.org/10.1007/978-3-642-12334-4_8. Visited on May, 2017. Cited at pag. 19

Gaye *et al.*(**2003**) Lalya Gaye, Ramia Mazé and Lars E. Holmquist. Sonic city: The urban environment as a musical interface. Em *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 109–115, Montreal, Canada. URL http://www.nime.org/proceedings/2003/nime2003_109.pdf. Visited on May, 2017. Cited at pag. 6

Gaye *et al.*(**2006**) Lalya Gaye, Lars E. Holmquist, Frauke Behrendt and Atau Tanaka. Mobile music technology: Report on an emerging community. Em *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 22–25, Paris, France. URL http://www.nime.org/proceedings/2006/nime2006_022.pdf. Visited on May, 2017. Cited at pag. 6

Geiger(2003) Günter Geiger. Pda: Real time signal processing and sound generation on handheld devices. Em *Proceedings of International Computer Music Conference (ICMC)*, pages 1–4, Singapore. National University of Singapore and Yong Siew Toh Conservatory of Music, International Computer Music Association. Cited at pag. 6, 10, 12

Gere(2006) Charlie Gere. *Art, time and technology*. Berg. Cited at pag. 7

Glinsky(1992) Albert Vincent Glinsky. *The theremin in the emergence of electronic music*. Phd Thesis, New York University. Cited at pag. 72

Gopinath and Stanyek(2014a) Sumanth Gopinath and Jason Stanyek. *The oxford handbook of mobile music studies*, volume 1. Oxford University Press. Cited at pag. 5, 6

Gopinath and Stanyek(2014b) Sumanth Gopinath and Jason Stanyek. *The oxford handbook of mobile music studies*, volume 2. Oxford University Press. Cited at pag. 6

GSMArena(2016) GSMArena. Nokia n95 vs. samsung galaxy s7 - gsmarena.com. http://www.gsmarena.com/compare.php3?idPhone1=1716&idPhone2=7821, 2016. Visited on May, 2017. Cited at pag. xiii, 59, 60

GSMArena(2017) GSMArena. Lg g pro lite dual vs. samsung i9300 galaxy s iii vs. sony xperia z3 compact - gsmarena.com. http://www.gsmarena.com/compare.php3?idPhone1=5736&idPhone2=4238&idPhone3=6538, 2017. Visited on May, 2017. Cited at pag. xiii, xiv, 29, 63, 64

Gupta(2013) Naresh Gupta. *Inside Bluetooth low energy*. Artech house. Cited at pag. 20

Harker *et al.*(2008) Alex Harker, Angie Atmadjaja, Jethro Bagust and Ambrose Field. Worldscape laptop orchestra: Creating live, interactive digital music for an ensemble of fifty performers. Em *Proceedings of the International Computer Music Conference (ICMC)*, Belfast, Northern Ireland. Sonic Arts Research Centre, Queen's University Belfast, International Computer Music Association. Cited at pag. 22

Hindle(2013) Abram Hindle. SWarmed: Captive portals, mobile devices, and audience participation in multi-user music performance. Em *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 174–179, Daejeon, Republic of Korea. Graduate School of Culture Technology, KAIST. URL http://nime.org/proceedings/2013/nime2013_62.pdf. Visited on May, 2017. Cited at pag. 14, 15

Hindle(2014) Abram Hindle. Cloudorch: A portable soundcard in the cloud. Em *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 277–280, London, United Kingdom. Goldsmiths, University of London. URL http://www.nime.org/proceedings/2014/nime2014_541.pdf. Visited on May, 2017. Cited at pag. 15

Hughes(2014) Neil Hughes. ipad art by singer bjork becomes first-ever app in moma's permanent collection - appleinsider. http://appleinsider.com/articles/14/06/13/ipad-art-by-singer-bjork-becomes-first-ever-app-in-momas-permanent-collection, 2014. Visited on May, 2017. Cited at pag. 6

IALO(2016) IALO. The international association of laptop orchestras, 2016. URL http://ialo.org. Visited on May, 2017. Cited at pag. 8

Ito *et al.*(2015) Akinori Ito, Kengo Watanabe, Genki Kuroda and Kenichiro Ito. isupercolliderkit: A toolkit for ios using an internal supercollider server as a sound engine. Em *Looking Back, Looking Forward: Proceedings of the 41st International Computer Music Conference, ICMC 2015, Denton, TX, USA, September 25 - October 1, 2015*. International Computer Music Association. Cited at pag. 10, 13

John(2013) David John. Updating the classifications of mobile music projects. Em *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 301–306, Daejeon, Republic of Korea. Graduate School of Culture Technology, KAIST. URL http://nime.org/proceedings/2013/nime2013_273.pdf. Visited on May, 2017. Cited at pag. 6

Jordà(1999) Sergi Jordà. Faust music on line: An approach to real-time collective composition on the internet. *Leonardo Music Journal*, 9:5–12. Cited at pag. 7, 8

Jourdain(1997) Robert Jourdain. *Music, the brain, and ecstasy: How music captures our imagination*. W. Morrow New York. Cited at pag. 8

Kim and Essl(2011) Jong Wook Kim and Georg Essl. Concepts and practical considerations of platform-independent design of mobile music environments. Em *Proceedings of the International Computer Music Conference (ICMC)*, pages 726–729, University of Huddersfield, England. Centre for Research in New Music (CeReNeM), International Computer Music Association. Cited at pag. 29

Kleimola(2015) Jari Kleimola. Daw plugins for web browsers. Em *1st Web Audio Conference, IRCAM, Paris, France*. Cited at pag. 11

Kon and Iazzetta(1998) Fabio Kon and Fernando Iazzetta. Internet music: Dream or (virtual) reality? Em *Proceedings of the Brazilian Symposium on Computer Music*, pages 69–81. Cited at pag. 8

Krekovic *et al.***(2015)** Miranda Krekovic, Gordan Krekovic and Franco Grbac. Sound my vision: Real-time video analysis on mobile platforms for controlling multimedia performances. Em *Proceedings of the International Conference on Sound and Music Computing (SMC)*, pages 235–240. National University of Ireland Maynooth. Cited at pag. 6

Lago and Kon(2004) Nelson Posse Lago and Fabio Kon. The Quest for Low Latency. Em *Proceedings of the International Computer Music Conference (ICMC)*, pages 33–36, Coral Gable, FL, USA. Frost School of Music, University of Miami, International Computer Music Association. Cited at pag. 8, 14, 22

Lazzarini *et al.***(2014)** Victor Lazzarini, Edward Costello, Steven Yi and John Fitch. Csound on the web. Em *Proceedings of the Linux Audio Conference*, Karlsruhe, Germany. ZKM. Cited at pag. 13

Lee and Freeman(2013) Sang Won Lee and Jason Freeman. echobo : Audience participation using the mobile music instrument. Em *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 450–455, Daejeon, Republic of Korea. Graduate School of Culture Technology, KAIST. URL http://nime.org/proceedings/2013/nime2013_291.pdf. Visited on May, 2017. Cited at pag. 14, 83

Lee and Freeman(2013) Sang Won Lee and Jason Freeman. Real-Time Music Notation in Mixed Laptop–Acoustic Ensembles. *Computer Music Journal*, 37(4):24–36. Cited at pag. 8

Lee *et al.***(2014)** Sang Won Lee, Georg Essl and Z. Morley Mao. Distributing mobile music applications for audience participation using mobile ad-hoc network (MANET). Em *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 533–536, London, United Kingdom. Goldsmiths, University of London. URL http://www.nime.org/proceedings/2014/nime2014_546.pdf. Visited on May, 2017. Cited at pag. 13

Lee *et al.***(2016)** Sang Won Lee, Antonio Deusany de Carvalho Jr and Georg Essl. Crowd in c [loud]: Audience participation music with online dating metaphor using cloud service. Em *Web Audio Conference*, Atlanta, Georgia, USA. Georgia Institute of Technology. Cited at pag. 15, 55, 84

Lehr and McKnight(2003) William Lehr and Lee W McKnight. Wireless internet access: 3g vs. wifi? *Telecommunications Policy*, 27(5–6):351 – 370. ISSN 0308-5961. URL http://www.sciencedirect.com/science/article/pii/S0308596103000041. Compeitition in Wireless: Spectrum, Service and Technology Wars. Cited at pag. 19

Levin(2001) Golan Levin. Dialtones - a telesymphony, september 2001. http://www.flong.com/projects/telesymphony, 2001. Visited on May, 2017. Cited at pag. 5

Levin(2004) Golan Levin. An informal catalogue of mobile phone performances, installations and artworks, 2001-2004. http://www.flong.com/texts/lists/mobile_phone, 2004. Visited on May, 2017. Cited at pag. 5

LINK(2017) TP LINK. Ac1750 wireless dual band gigabit router tp link. http://www.tp-link.com/us/products/details/cat-9_Archer-C7.html, 2017. Accessed February, 2017. Cited at pag. xiv, 65

Ma *et al.***(2013)** Zhizhong Ma, Yuansong Qiao, Brian Lee and Enda Fallon. Experimental evaluation of mobile phone sensors. Em *Signals and Systems Conference (ISSC 2013), 24th IET Irish*, pages 1–8. IET. Cited at pag. 35

Matuszewski *et al.***(2016)** Benjamin Matuszewski, Norbert Schnell and Samuel Goldszmidt. Interactive audiovisual rendering of recorded audio and related data with the wavesjs building blocks. Em *Web Audio Conference*, Atlanta, GA, USA. Georgia Institute of Technology. Cited at pag. 11

McCartney(2002) James McCartney. Rethinking the computer music language: Supercollider. *Computer Music Journal*, 26(4):61–68. Cited at pag. 13

Miller and Helft(2009) Claire Cain Miller and Miguel Helft. From pocket to stage, music in the key of iphone - nytimes.com. http://www.nytimes.com/2009/12/05/technology/05orchestra.html, 2009. Visited on May, 2017. Cited at pag. 6

Miranda and Wanderley(2006) Eduardo Reck Miranda and Marcelo M Wanderley. *New digital musical instruments: control and interaction beyond the keyboard*, volume 21. AR Editions, Inc. Cited at pag. 11, 12

Mogul and Deering(1990) Jeffrey Mogul and Steve Deering. Rfc1191: Path mtu discovery. *Internet Engineering Task Force*. Cited at pag. 35

Ogborn(2014) David Ogborn. Live coding in a scalable, participatory laptop orchestra. *Computer Music Journal*, 38(1):17–30. Cited at pag. 9

Padre(2017) Joe Padre. Understanding touch responsiveness - touchscreen technology series 2 - developer world. https://developer.sonymobile.com/2014/07/02/understanding-touch-responsiveness-touchscreen-technology-series-2/, 2017. Visited on May, 2017. Cited at pag. 35, 74

Piquemal and McCormick(2017) Sebastien Piquemal and Chris McCormick. Webpd. https://github.com/sebpiq/WebPd, 2017. Visited on May, 2017. Cited at pag. 12

PubNub(2017) PubNub. Realtime apps made simple | pubnub. http://www.pubnub.com, 2017. Visited on May, 2017. Cited at pag. xiii, 32

Puckette(1997) Miller Smith Puckette. Pure Data. Em *Proceedings of the International Computer Music Conference*, pages 224–227. International Computer Music Association. Cited at pag. 12

Pusher(2017) Pusher. Pusher | leader in reatime technologies. http://www.pusher.com, 2017. Visited on May, 2017. Cited at pag. xiii, 32

Raman and Chebrolu(2007) B. Raman and K. Chebrolu. Experiences in using wifi for rural internet in india. *IEEE Communications Magazine*, 45(1):104–110. ISSN 0163-6804. Cited at pag. 21

Rice and Hay(2010) Andrew Colin Rice and Simon Hay. Decomposing power measurements for mobile devices. Em *PerCom*, volume 10, pages 70–78. Cited at pag. 22

Riley(1964) Terry Riley. In c. Composition, 1964. Cited at pag. 83

Roberts and Kuchera-Morin(2012) Charlie Roberts and JoAnn Kuchera-Morin. Gibber: Live coding audio in the browser. Em *Proceedings of the International Computer Music Conference (ICMC)*, Ljubljana, Slovenia. IRZU - the Institute for Sonic Arts Research, International Computer Music Association. Cited at pag. 9, 11, 87

Rohs and Essl(2007) Michael Rohs and Georg Essl. CaMus 2: collaborative music performance with mobile camera phones. Em *Proceedings of the International Conference on Advances in Computer Entertainment Technology*, pages 190–195. ACM. Cited at pag. 12

Rohs *et al.***(2006)** Michael Rohs, Georg Essl and Martin Roth. Camus: Live music performance using camera phones and visual grid tracking. Em *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 31–36, Paris, France. URL http://www.nime.org/proceedings/2006/nime2006_031.pdf. Visited on May, 2017. Cited at pag. 12

Rottondi *et al.***(2016)** Cristina Rottondi, Chris Chafe, Claudio Allocchio and Augusto Sarti. An overview on networked music performance technologies. *IEEE Access*. Cited at pag. 8

Schiavoni(2013) Flávio Luiz Schiavoni. *Medusa - Um ambiente musical distribuído*. Phd Thesis, Instituto de Matemática e Estatística, Universidade de São Paulo. Cited at pag. 9

Schiavoni and Queiroz(2012) Flávio Luiz Schiavoni and Marcelo Queiroz. Network distribution in music applications with medusa. Em *Proceedings of the Linux Audio Conference*, pages 9–14. Cited at pag. 9

Schiavoni *et al.***(2011)** Flávio Luiz Schiavoni, Marcelo Queiroz and Fernando Iazzetta. Medusa-a distributed sound environment. Em *Proceedings of the Linux Audio Conference*, pages 149–156. Cited at pag. 9, 27

Schiavoni *et al.***(2013)** Flávio Luiz Schiavoni, Marcelo Queiroz and Marcelo Wanderley. Network music with medusa: A comparison of tempo alignment in existing midi apis. Em *Proceedings of the Sound and Music Computing Conference, Stockholm, Sweden*, volume 91. Cited at pag. 9

Schiemer and Havryliv(2005) Greg Schiemer and Mark Havryliv. Pocket gamelan: a pure data interface for mobile phones. Em *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 156–159, Vancouver, BC, Canada. URL http://www.nime.org/proceedings/2005/nime2005_156.pdf. Visited on May, 2017. Cited at pag. 6, 10, 12

Schuett(2002) Nathan Schuett. The effects of latency on ensemble performance. Undergraduate honors thesis, CCRMA Stanford Center for Computer Research in Music and Acoustics, Stanford, CA, USA. Cited at pag. 8

Shannon(1948) Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal, The*, 27(3):379–423. ISSN 0005-8580. Cited at pag. 18

Shannon *et al.***(1961)** Claude E Shannon *et al.* Two-way communication channels. Em *Proc. 4th Berkeley Symp. Math. Stat. Prob*, volume 1, pages 611–644. Citeseer. Cited at pag. 18

Smith and Doward(2010) Caspar Llewellyn Smith and Jamie Doward. Gorillaz give away their new album made on an ipad - the guardian. https://www.theguardian.com/music/2010/dec/25/damon-albarn-fall-gorillaz-ipad, 2010. Visited on May, 2017. Cited at pag. 6

Stolfi *et al.***(2017a)** Ariane Stolfi, Mathieu Barthet, Fabio Gorodscy and Antonio Deusany de Carvalho Junior. Open band: A platform for collective sound dialogues. Em *Proceedings of the 12th International Audio Mostly Conference on Augmented and Participatory Sound and Music Experiences*. ACM. Cited at pag. 56

Stolfi *et al.***(2017b)** Ariane Stolfi, Mathieu Barthet, Fabio Gorodscy, Antonio Deusany de Carvalho Junior and Fernando Iazzetta. Open band: Audience creative participation using web audio synthesis. Em *Proceedings of the 3rd Web Audio Conference*. Cited at pag. 56

Swift *et al.***(2014)** Ben Swift, Henry Gardner and Andrew Sorensen. Networked livecoding at vl/hcc 2013. Em *Visual Languages and Human-Centric Computing (VL/HCC), 2014 IEEE Symposium on*, pages 221–222. IEEE. Cited at pag. 9

Tahiroglu(2009) Koray Tahiroglu. Towards an experimental platform for collective mobile music performance. Em *Proceedings of the International Conference on Sound and Music Computing (SMC)*, pages 23–25. INESC Porto, the Research Center for Science and Technology in Art of the Universidade Católica Portuguesa in Porto, ESMAE and Casa da Música. Cited at pag. 6

Tanaka(2004) Atau Tanaka. Mobile music making. Em *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 154–156, Hamamatsu, Japan. URL http://www.nime.org/proceedings/2004/nime2004_154.pdf. Visited on May, 2017. Cited at pag. 6

Tanzi(2001) Dante Tanzi. Observations about music and decentralized environments. *Leonardo*, 34(5):431–436. Cited at pag. 8

The Android Open Source Project(2016a) The Android Open Source Project. Android hardware abstraction layer: hardware/libhardware/include/hardware/sensors.h source file. https://source.android.com/devices/halref/sensors_8h_source.html, 2016a. Visited on May, 2017. Cited at pag. xiii, 61

The Android Open Source Project(2016b) The Android Open Source Project. Sensor types | android open source project. https://source.android.com/devices/sensors/sensor-types.html, 2016b. Accessed June, 2016. Cited at pag. xiii, 61

Tomiyoshi(2013) Marcio Masaki Tomiyoshi. *Performances musicais distribuídas através de Internet residencial*. Master Thesis, Instituto de Matemática e Estatística, Universidade de São Paulo, Brasil. Cited at pag. 9

Trueman(2007) Dan Trueman. Why a laptop orchestra? *Organised Sound*, 12(02):171–179. Cited at pag. 8

Vercoe and Ellis(1990) Barry Vercoe and Dan Ellis. Real-time csound: Software synthesis with sensing and control. Em *Proceedings of the International Computer Music Conference (ICMC)*, volume 90, pages 209–211, Glasgow, Scotland. Univ. of Glasgow, International Computer Music Association. Cited at pag. 13

Wang(2014) Ge Wang. Ocarina: Designing the iphone's magic flute. *Computer Music Journal*, 38(2):8–21. Cited at pag. 5, 11, 13

Wang *et al.***(2003)** Ge Wang, Perry R Cook *et al.* Chuck: A concurrent, on-the-fly, audio programming language. Em *Proceedings of the International Computer Music Conference (ICMC)*, Singapore. National University of Singapore and Yong Siew Toh Conservatory of Music, International Computer Music Association. Cited at pag. 13

Wang *et al.***(2008)** Ge Wang, Georg Essl and Henri Penttinen. Do mobile phones dream of electric orchestras. Em *Proceedings of the International Computer Music Conference (ICMC)*, pages 1–8, Belfast, Northern Ireland. Sonic Arts Research Centre, Queen's University Belfast, International Computer Music Association. Cited at pag. 6, 10, 13, 14

Weinberg(2003) Gil Weinberg. *Interconnected Musical Networks – Bringing Expression and Thoughtfulness to Collaborative Music Making*. PhD Thesis, Media Lab, Massachusetts Institute of Technology, USA. Cited at pag. 7, 8, 9

Weinberg(2005) Gil Weinberg. Interconnected musical networks: Toward a theoretical framework. *Computer Music Journal*, 29(2):23–39. Cited at pag. 5, 9, 83

Weitzner *et al.*(2012) Nathan Weitzner, Jason Freeman, Stephen Garrett and Yan-Ling Chen. massmobile -an audience participation framework. Em *Proceedings of the International Conference on New Interfaces for Musical Expression*, Ann Arbor, Michigan. University of Michigan. URL http://www.nime.org/proceedings/2012/nime2012_128.pdf. Visited on May, 2017. Cited at pag. 10, 14, 15

Wyse and Subramanian(2013) Lonce Wyse and Srikumar Subramanian. The viability of the web browser as a computer music platform. *Computer Music Journal*, 37(4):10–23. Cited at pag. 10

Yi and Lazzarini(2012) Steven Yi and Victor Lazzarini. Csound for android. Em *Linux Audio Conference*, volume 6, pages 29–34, Center for Computer Research in Music and Acoustics (CCRMA), Stanford University, California, USA. Cited at pag. 10, 74