Computer Science Tripos – Part II – Project Proposal

A Distributed Approach to Collaborative Live Music Performances

Elias Calocane, King's College

Originators: Elias Calocane, Sam Aaron and Martin Kleppmann

13 October 2017

Project Supervisor: Martin Kleppmann

Project Adviser: Sam Aaron

Director of Studies: Dr T. G. Griffin

Project Overseers: M. P. Fiore & T. M. Jones

Introduction

Making music in a documented fashion using computers is a well-established process. Countless Digital Audio Workstations (DAWs) are available for users to create music on their machines using components such as real and virtual instruments, vocal tracks and other types of samples. In terms of collaborating, DAWs allow multiple users to work on a project, although it is convention for this to be done in the same location and machine – further flexibility is however possible, as users are also able to collaborate by sharing project files.

In terms of composition of music in a fashion through which users collaborate remotely (for example over a network), the options are far more limited. Ableton created Link, functionality for their DAW and other Link-compatible software. Link allows musicians using compatible software to play in sync (live) whilst connected to the same local network – by ensuring that all players adhere to a global sense of time through tempo synchronisation, beat alignment and phase synchronisation. This project sets out to overcome the local network restrictions that are part of Link and provide a Link-inspired functionality that works in a wide area network, whilst still focusing on the element of live performances by making use of Sonic Pi – thus providing a distributed method for musicians to collaborate.

A wide area approach naturally introduces challenges involving dealing with networking issues such as latency and jitter, as well as (very critically) the problem of making sure everyone involved in the performance observes the same events in the same order. This distributed sense of coherence is vital to ensuring that everyone perceives the performance in the same way (so that it sounds the same for all users) and is the main focus of this project.

Starting point

Relevant Courses include Networking and Distributed/Concurrent Systems from Part IB.

Access to Ableton Link's developer resources (including GitHub repository with source code).

Access to the source code for Sonic Pi through GitHub.

Resources required

I will be primarily relying on the use of my own laptop for the project – a laptop with an Intel Core i7 4720HQ Processor, 16GB RAM, 512GB SSD and NVIDIA GeForce GTX 960M GPU running Windows 10.

If my laptop should fail, I will use a MCS workstation in the Computer Laboratory.

I will run weekly backups to a personal USB stick as well as to GitHub and Google Drive.

I will require access to a VM Server in order to connect Sonic Pi instances (running on different machines) to each other.

Key Concepts

These are the main concepts/technologies this project will try to bring together

Link

Link allows users to play in sync by synchronising tempo, aligning beats and synchronising phase.

Tempo synchronisation

Tempo can be described as the rate at which a musical piece progresses, measured in beats per minute (BPM). In order for different musicians to play with each other in a coherent manner, it is essential that their contributions to the musical piece advance at the same rate, ~~simply~~ making sure that all musicians adhere to the same predefined tempo.

Beat alignment

Advancing at the same rate is not sufficient for contributions to a specific piece to sound coherent – it is also essential that the beats occur at the same exact time (that they are aligned). In order to clarify what is meant by this, we adopt a numbering notion for beats such that the first beat is denoted by number 1, the second by number 2, the third by number 3, etc. Beat alignment ensures that the difference between the beats of any two users, is always an integer, such that it is possible for one user to be at beat 4 whilst another is at beat 2, but it is not possible for third user to be at beat 2.5 (halfway between beat 2 and 3).

Phase synchronisation

When dealing with loops or bars (consisting of a set number of beats) it is also useful to ensure that there is some consensus regarding the points at which they start and end. This is so that loops created by different users start/end at sensible places relative to each other. For loops of the same size, this means starting and finishing together (which may involve delaying the triggering of loops so their start in sync with others). For loops of different sizes, there may be more than one point at which a loop can be triggered – for example a 4-beat loop can start either at the start or the middle of an 8-beat loop.

Open Sound Control (OSC)

On networks, OSC messages are most commonly transported as UDP packets.

Open Sound Control defines a message-based protocol that allows communication between different multimedia devices such as computers and synthesisers. OSC messages can be used to encode descriptions of sounds (or musically related events) that can then be used in various scenarios. We will make use of OSC messages in order to communicate between Sonic Pi instances (being run on different machines by different users) in order to ensure that they all agree on which sounds to produce.

Sonic Pi

Sonic Pi defines a well-established language and software that allows users to produce music through live-coding. OSC messages are used within the software in order to describe music related events that need to be relayed to a speaker (through an audio synthesis platform called SuperCollider, integrated into Sonic Pi). Sonic Pi also defines temporal semantics which provide ways of reasoning about when sounds produced are by the software as well as making sure this occurs in a deterministic manner (or relaying to the user when that is not a possibility).

Structure

In the current implementation of Sonic Pi, these OSC messages flow only between the Sonic Pi process and SuperCollider running on the same machine. In this project I plan to extend Sonic Pi for distributed use by also transmitting these OSC messages over wide-area networks to other participants in the musical session.

Implementation

A user ~~will produce~~ writes a script (using Sonic Pi) that ~~will in order~~ generates OSC messages describing the sound they intend to make. ~~These~~ said messages are then interpreted ~~within~~ the SuperCollider synth to ~~relay~~ play the correct sound ~~to~~ from the user's speakers. These messages need to be sent from a Sonic Pi instance to all other instances that are part of the performance so that other contributors are also able to hear the sounds that the original user intended to produce. A server will be used to setup a connection between Sonic Pi instances so that OSC messages can be transported between them.

Since we are focusing on a collaborative performance, it is vital that all users hear exactly the same sounds – or in other words, that the lists of OSC messages interpreted by each user's SuperCollider synth, are exactly the same. It is therefore important to account for aspects such as latency and jitter in the network to ensure that we are able to provide a deterministic order of events for all users when they are all transmitting OSC messages to each other.

Evaluation

Mention that Sonic Pi schedules sounds ahead of time (by default 0.5s), which gives you time to relay the OSC messages over the network.
Explain what happens when several users are playing: the messages sent to SuperCollider are the union of all the notes generated by the participants. Sort OSC messages by scheduled timestamp, so everyone should see same message sequence.

Be more scientific! What hypothesis are you testing? (Coherent sound can be achieved if network latency/jitter below some threshold?) What will you measure? (Latency, packet loss rate – on university networks, and home ISPs?) How will you improve your software based on the measurements? (eg. automatically tuning schedule-ahead time to network conditions? Measure whether TCP or UDP give you better results?)

After setting up a scenario where at least 3 users are connected via the server, we will examine the list of OSC messages that each user's SuperCollider synth receives and see whether that said lists are identical for all users.

We then aim to focus on the deviation between the timelines of the users - even though users may be hearing the same thing, the events are unlikely to occur at the exact same points in real time. The idea is to find the limit at which the deviation is too great for a suitable interactive experience (so that users can react to each other's contributions).

We also aim to find the point at which our idea of determinism regarding the distributed consensus of events breaks down. We want to investigate the point at which network issues such as latency are too great for messages to be scheduled at the same time for all users and thus, under what conditions we are unable to provide a deterministic view of events across users.

The last two parts of the evaluation can be carried out using dummynet (or a similar tool) to emulate poor network conditions (controlling packet delay, loss, etc.).

Success citeria

Fully implement a way for at least three users to setup a connection and play together using Sonic Pi.

Ensure determinism/coherence of each user's notion of the performance (when network conditions are acceptable) so all users hear the same thing.

Ensure that users' timelines don't drift so far apart that they are unable to react to each other's contributions (even when determinism holds).

Possible extensions

If I achieve my main result early I shall try the following,

Integrate WebRTC into the project instead of using a server to provide a peer-to-peer implementation – if such an approach were to reduce latency significantly, it may be possible to implement features such as making it possible for different users to update (different) aspects of the same sound at the same time (whilst still keeping a deterministic view of events across all users). Carrying out the evaluation steps for this new implementation would allow for a comparison between the original server-based and peer-to-peer approaches to the project.

Trying out different schemes for getting coherence.

Timetable

1. **Michaelmas weeks 2–4** Read up on similar projects already out there and familiarise myself with Ableton Link's and Sonic Pi's source code

Before you get into WebRTC, some things I suggest trying:
-Latency and packet loss measurements on different networks, so you know what your limits are
-Experiment whether TCP or UDP gives you a better (more coherent) performance (while still going via the server)

2. **Michaelmas weeks 5–6** Start first implementation of the server.
3. **Michaelmas weeks 7–8** Finish the first implementation of the server - by this point I aim to be able to setup a connection between users as well as send, receive OSC messages and deal with superficial synchronisation of various nodes (via NTP for example)
4. **Michaelmas vacation** If previous stages are fully complete, try to implement alternative method of communication via WebRTC. Start second implementation of the server.
5. **Lent weeks 0–2** Start writing progress report. Finish the second implementation of the project - by this point I aim to be able to provide some sense of determinism/consensus across all users
6. **Lent weeks 3–5** Run main experiments and achieve working project.
7. **Lent weeks 6–8** Start planning Dissertation. Absorb any incomplete work from the previous stages (if any of them were delayed). If possible, carry on/start integration of alternative WebRTC-based method of communication.
8. **Easter vacation:** Working on the extensions of the project and writing the main chapters of the dissertation.
9. **Easter term weeks 0–2:** Carry out any further evaluation and complete dissertation.
10. **Easter term week 3:** Proof read and submit dissertation - assuming no considerable delays were experienced in the previous stages.

I would recommend starting dissertation writing much earlier. Aim to have a first draft of dissertation by start of Easter vacation, so that you can use vacation to revise for your finals! You can always add an extension to a dissertation draft later. So I would first write the diss, then get into extensions.