Breaking Links: Evaluating distributed time synchronization for musical applications using Ableton Link

Xavier Riley

Submitted for the Degree of Master of Science in

Distributed and Networked Systems

Department of Computer Science
Royal Holloway University of London
Egham, Surrey TW20 0EX, UK

August 6, 2018

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count:

Student Name:

Date of Submission:

Signature:

Abstract

With the recent explosion of connected musical devices, the challenge of making these play "in time" with each other mounts against application developers and device manufacturers. Ableton Link[1] aims to provide robust, resilient musical synchronization using principles from distributed systems programming, in contrast to previous master/slave approaches. However the evaluation criteria for such a system are not well represented in the existing literature, with particular reference to a musical context. The following presents a system for empirical testing of the Ableton Link library using the Jepsen[2] testing framework, along with a set of criteria for evaluting similar libraries that may be developed in future.

Contents

1 Introduction

The synchronizing of events is fundamental to our perception of musical performance. Where performers are using networked connected devices, the challenge incorporates the well studied problem of clock synchronization from distributed systems.

As network technologies continue to grow in usage and importance to musical performances[3], it becomes increasingly important to find common approaches to allow devices and applications to synchronize without reliance on proprietary protocols. Music programming environments in the academic space are well catered for in this regard, however applications in the consumer market have tended to lag behind advances in technology. Either the production of music is limited to a single device, or additional devices are synchronized using specialized hardware dedicated to synchronizing frames.

1.1 Fundamental problems of music synchronization

* getting clocks in sync * keeping them in sync (drift, latency) * network bandwidth (commodity network hardware, high number of devices) * fault tolerance (master availability, *musical* consistency, convergence time) * ease of setup and deployment

1.2 About Ableton Link

Ableton Link aims to solve some of these issues by supplying an open source, permissively licensed C++ library for integration with application code. As well as the popular digital audio workstation Ableton Live, implementations exist for a large number of mobile applications and for many of the popular music programming environments.

It differs from existing approaches in that it does not rely on a master process to propagate timing information directly. Instead nodes will establish a session, using a reference to the start time of the oldest member of the group even if that member is no longer present.

Clock synchronization is performed using a Kalman filter[] which adds a level of robustness to jitter introduced by the network along with a more accurate reflection of the real delay. This approach is relatively sophisticated when compared with others using Cristian algorithm.

Integration is handled by application developers who are left to integrate a small API. C++ has widespread support for integration with many popular languages, making this viable for the majority of existing applications.

Setup for the end user is virtually transparent - network discovery takes place automatically on all interfaces. Clock synchronization is performed automatically for nodes joining a session and then at 60 second intervals afterwards. Simple transport commands (start, stop) and tempo changes are also propagated automatically by reliable broadcast.

Failures, drop outs and re-entry are all handled with a model of eventual consistency where last-write-wins takes effect for changes in tempo and transport state.

2 Background research

The use of networks in computer music is an active area of study, with much of the research being driven by "laptop orchestras"[5] centered around academic institutions. This has led to approaches centering around the Open Sound Control protocol[6][3][4] as the "lingua franca" of connected musical applications, although older methods include the use of MIDI, SMPTE and other standards - see Goltz[1] for a review of these.

2.1 Prior testing approaches

* in production/performance (Landini) * via hardware recording to sound file (PiGMI + Landini)

3 Criteria and design

3.1 Consistency in a musical context

Convergence time as a percentage of the session

Longest time for period of divergence.

3.2 Accuracy versus Bandwidth

3.3 Use of Jepsen, Docker and log parsing

Jepsen desgined to test linearizability through fault injection, however it is flexible.

Docker allows straightforward reproducbility on development machines or in CI testing.

Use of debug logs in Link

3.4 Topologies and latencies

Easy (connected, zero delay) Medium (Ring? bridge? zero delay) Hard (Line, zero delay)

ditto small delay

ditto large, variable or uneven delay

4 Experimental results

Results here, maybe detailed graphs in appendix

5 Analysis

Resilient to different network conditions. Usually converges.

5.1 Identifying and triggering invalid states

6 Further work

Generalize the work to other solutions.

Create CI testing for the Link library.

Suggest improvements to Link - Google nanosecond precision paper. Potential uncertainty window around synchronization events. Configuration/visibility of session information.

Integration into Sonic Pi. Windows compatibility.

7 Conclusions

References

[1] Florian Goltz. Ableton link–a technology to synchronize music software. In *Linux Audio Conference 2018*, page 39, 2018.

[2] Kingsbury, Kyle. Jepsen - a framework for distributed systems verification, with fault injection, 2018. [Online; accessed 6-August-2018].

[3] Sebastian OH Madgwick, Thomas J Mitchell, Carlos Barreto, and Adrian Freed. Simple synchronisation for open sound control. 2015.

[4] Jascha Narveson and Dan Trueman. Landini: a networking utility for wireless lan-based laptop ensembles. In *Proc. SMC Sound, Music and Computing Conference*, 2013.

[5] Dan Trueman. Why a laptop orchestra? *Organised Sound*, 12(2):171–179, 2007.

[6] Matthew Wright. Open sound control: an enabling technology for musical networking. *Organised Sound*, 10(3):193–200, 2005.

A Self assessment

B Running the tests

C Configuring Jepsen to support other systems and tests

D Code listings

E Professional issues

F Acknowledgements

Open source code Colleagues at Heroku Salesforce for funding Gregory and staff at RHUL Em