# Networked Livecoding at VL/HCC 2013

Ben Swift, Henry Gardner
Research School of Computer Science
Australian National University
Email: ben.swift@anu.edu.au

Andrew Sorensen
Institute for Future Environments
Queensland University of Technology
Email: a.sorensen@qut.edu.au

***Abstract*—Network connectivity offers the potential for a group of musicians to play together over the network. This paper describes a trans-Atlantic networked musical livecoding performance between Andrew Sorensen in Germany (at the Schloss Daghstuhl conference on *Collaboration and Learning through Live Coding*) and Ben Swift in San Jose (at *VL/HCC*) in September 2013. In this paper we describe the infrastructure developed to enable this performance.**

## I. A networked livecoding architecture

The networked performance at VL/HCC 2013 involved two artists: Ben Swift in San Jose, California and Andrew Sorensen in Germany, as well as an audience at each location. Both artists were livecoding using the Extempore programming language [3].

Although audio and especially video streams require high-bandwidth connections, advances in the technology and economics of such connections have made the prospect of networked musical performance (NMP) increasingly attainable in recent years [1].

In general, a NMP system can choose to use either a "signal-over-the-wire" (SotW) or a "control-over-the-wire" (CotW) approach [4]. In a SotW system, each musician generates an audio signal, and the NMP system streams each of these audio signals over the network. In the CotW approach, lightweight control data (such as MIDI [5], [6] or even source code [7]) is sent over the network, and this data is turned into an audio signal at each end using an appropriate synthesis engine.

The VL/HCC performance setup used the CotW approach. The control data—in the form of the source code written by the livecoding artists at each end—was sent over the network to be evaluated independently at both ends as shown in Figure 1.

Code sharing was performed at an editor level through a custom Emacs [8] plugin. While the collaborative musical performance could have taken place if the code was not shared (so that each audience could only see their local livecoder's code) there is a strong commitment in the livecoding community to project the code for the audience. Because of this, the audiences at each end could see and hear both livecoders together, in order to appreciate the full extent of the live musical collaboration over the network.

The performance took place over a hotel internet connection in San Jose, through an SSH tunnel to the Dagstuhl conference centre in Germany. The SSH tunnel through Australia was used to provide a public IP address for each artist to connect to, which is often difficult to obtain on a hotel network due to network address translation (NAT). By forwarding the appropriate ports on this server (which had a known global IP), we were able to ensure that the connection between the computers could be made regardless of the local network setup. The server was located in Australia for convenience (it is a server owned by Ben Swift). In principle a server which was closer to either endpoint could have been used, but in testing it was clear that the bandwidth and latency of this setup was acceptable.

All of the software involved in this networked performance is available on GitHub at https://github.com/digego/extempore under an MIT licence.

## II. Benefits and limitations of this networked livecoding architecture

In livecoding a great deal of the performance involves the triggering of code *asynchronously*: scheduling musical events at some future time based on beat and tempo. This allows the sidestepping of the round-trip latency problems associated with the SotW approach, since as long as the code arrives before it is scheduled, the result is perfect (local) synchrony. Musical events which are meant to happen simultaneously happen on the same audio sample, which allows collaboration not just on a beat level but even sample-level phasing effects reminiscent of Steve Reich.

Livecoding is an excellent candidate for a CotW architecture because the control data is so rich—the code being written by each livecoder during the performance is written in a Turing-complete programming language (in the case of this performance, this language was Extempore [3]). This is in contrast to a MIDI-based CotW approach. MIDI is in general limited to the control of a few specific parameters such as pitch, velocity, and duration. In livecoding, however, the remote livecoder can write create a program to generate *any* audio stream at the other end.

From an information-theoretic perspective, sending only pitch/velocity/duration over the wire (as in MIDI) means that there needs to be sophisticated infrastructure at the receiving end, for example a software synthesizer or sampler. In the livecoding CotW paradigm, the livecoder can send the whole synthesizer over the wire as source code, as well as the subsequent pitch/velocity/duration commands to play it. This gives the networked performers a great deal of control over the audio stream generated at each end, while retaining the low bandwidth requirements of CotW networked performance.

These techniques were used by Ben and Andrew in the VL/HCC–Dagstuhl performance, where Extempore's support

978-1-4799-4035-6/14/$31.00 ©2014 IEEE

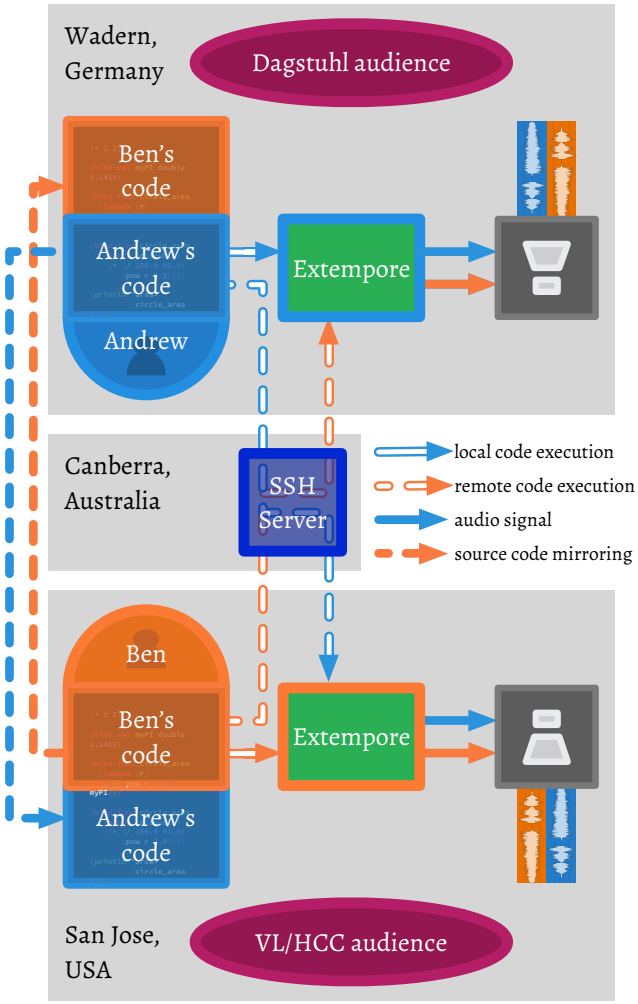


Fig. 1. The VL/HCC networked livecoding architecture. Both Andrew and Ben performed using the Extempore livecoding software package, and each artist was connected "live" to both their local Extempore process and the remote process at the other location, through an SSH tunnel in Canberra, Australia. In this way, each artist had their code evaluated both locally *and* remotely, and the audio signal was rendered at each location.

for real-time sample-level audio manipulation and signal processing offered the the freedom to work at a MIDI-like "note level" or a lower "signal level" as desired.

There are some limitations to the NMP approach we used for the VL/HCC performance. The most challenging one is dealing with different data and computational environments at each end. Since each artist's code was evaluated in both their local and the remote Extempore process simultaneously, then these environments need to be as similar as possible. If a particular piece of code was evaluated successfully at one location but triggered an error at the other (for example due to a missing audio sample file at one end) then the executed code will have had different results at each site. Similarly, if the performance is happening on a laptop at one end and a high-powered server at the other, then code which may run smoothly on the server may glitch and cause dropouts on the laptop. These challenges can be mitigated by ensuring that the system and computational environments are similar at all network endpoints.

The other limitations of this approach to networked livecoding are minor. The parallel execution means sacrificing some determinism, for example the pseudo-random number generator (PRNG) will in general give different results at each end. Again, it is possible to code defensively to deal with this limitation, only using these functions in situations where identical output at each end is not necessary.

## III. Conclusion

The networked performance at VL/HCC 2013 was a great success, both from a technical and artistic perspective. Mark Guzdial (Georgia Institute of Technology), a Dagstughl seminar attendee, described the performance as

> ...an amazing duet between Andrew Sorensen at Dagstuhl and Ben Swift at the VL/HCC conference in San Jose using their Extempore system. [2, p164]

The NMP architecture we describe here was designed specifically for livecoding—it is not easily applicable to a transatlantic jam session with conventional instruments. However, there is a growing class of interactive and collaborative networked performances where the combination of asynchronous musical scheduling and lightweight control data representation does allow for the CotW setup described in this paper.

We have continued to use the NMP software infrastructure developed for the Dagstuhl–VL/HCC performance in other networked performances. We hope to continue to refine these techniques to allow even more seamless and engaging networked livecoding performances in the future.

## References

[1] J. Lazzaro and J. Wawrzynek, *A case for network musical performance*. New York, New York, USA: ACM, Jan. 2001.

[2] A. Blackwell, A. McLean, J. Noble, and J. Rohrhuber, Eds., *Collaboration and learning through live coding*, ser. Dagstuhl Reports. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, Jan. 2014, vol. 3.

[3] A. Sorensen. Extempore. [Online]. Available: http://extempore.moso.com.au/

[4] A. Carôt, P. Rebelo, and A. Renaud, "Networked Music Performance: State of the Art," in *Audio Engineering Society International Conference*. Audio Engineering Society, 2007.

[5] A. Brown, S. Dillon, T. Kerr, and A. Sorensen, "Evolving Interactions: Agile design for networked media performance," *OZCHI '09 Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group: Design*, 2009.

[6] G. Hajdu, "Quintet.net: An Environment for Composing and Performing Music on the Internet," *dx.doi.org*, Mar. 2006.

[7] J. Rohrhuber, A. de Campo, R. Wieser, and J. van Kampen, "Purloined Letters and Distributed Persons," *Music in the Global Village Conference (Budapest)*, Jan. 2007.

[8] R. M. Stallman, "EMACS the extensible, customizable self-documenting display editor," *ACM SIGPLAN Notices*, vol. 16, no. 6, pp. 147–156, Jun. 1981.