Support for live collaborative distributed coding in Node-RED

Stephen Church

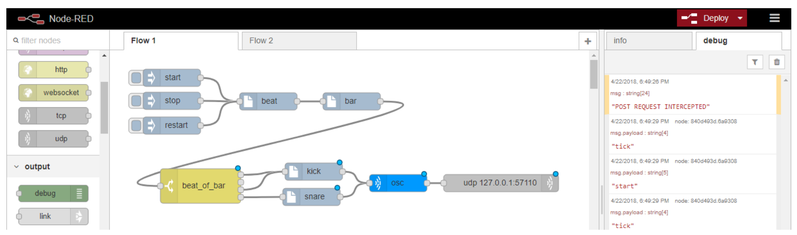
School of Engineering and Computing Sciences, Durham University

Introduction

Live coding is a method of developing and deploying source code in real-time to create a performance, usually related to music. A system has been created by Steven Bradley to allow live coding of music using Node-RED; a visual block-based programming environment. This project focused on extending this system to allow on-site collaboration in a distributed setting with the aim of allowing groups to create music performances together. The project was split into two main goals: synchronising beat generation and tempo between users; recording and replaying the performances.

Node-RED

Node-RED is a programming tool that allows users to develop simple Internet-Of-Things (IoT) applications by using boxes (nodes) and wires to connect them (to create flows); it facilitates rapid development and allows the creation of simple applications with minimal programming knowledge. Changes to flows and nodes are only applied when the system is deployed so it acts as a small-scale insight into a normal development process. During run-time, messages can be passed through the system using special Inject nodes to change the behaviour without the need to re-deploy the system.

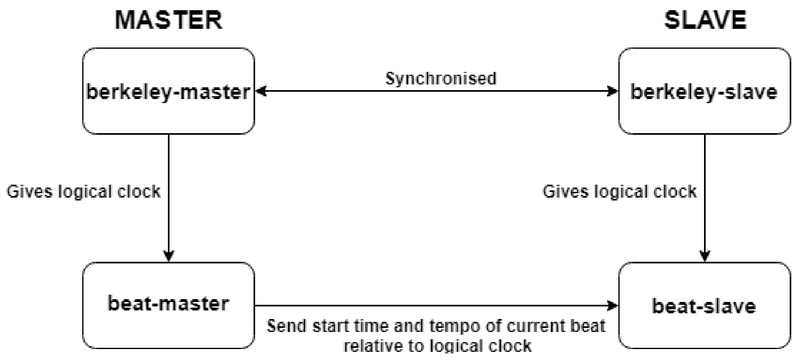


Berkeley nodes

The solution involved synchronising logical application clocks and using those to set the times of the beats for each user.
Berkeley nodes were created for Node-RED to synchronise the logical clocks between one master and multiple slaves. They communicate via UDP and set a Node-RED global "clock" variable to allow all other nodes to access the clocks.

Beat nodes

Modified original beat nodes to allow one to be a master and others to be slaves. These nodes communicate via UDP and use the synchronised logical clocks created by the berkeley nodes.



The diagram shows how the beat-master sends relevant information to the slaves so that the slaves can calculate the same beat timings relative to a synchronised logical clock. The tempo is also sent so that the slaves can update their own if necessary.

Recording

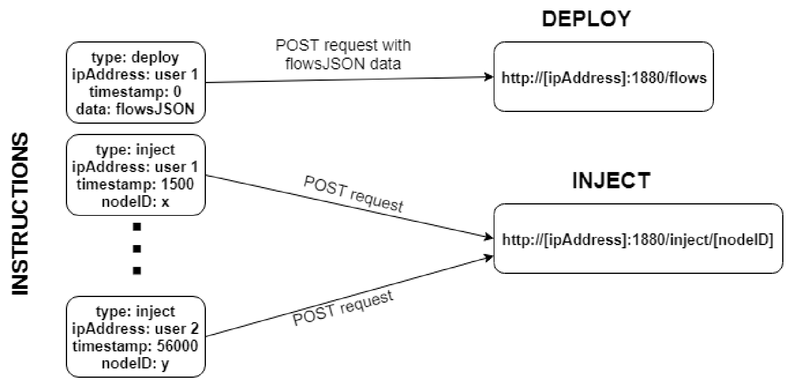
The aim was to record the steps taken in Node-RED to create the music performances without just recording audio; this allows other users to replay the exact steps taken during the performance, in order to analyse, reuse or even adapt the steps to change the final music generated.
Git is used for saving the steps involved in a Node-RED performance.
Two types of instructions are saved for each user:

- **Deployment instruction** - stores a copy of the flows (in JSON format) when re-deployed to give a snapshot of the system;
- **Inject instruction** - every time a message is injected into the system, it is saved as they control what happens within a given flow

All instructions are saved with a timestamp relative to the logical clocks defined by the berkeley nodes, so all instructions are saved in the correct order and each user has its own instruction file, named with an IP address.

Replaying

To replay the instructions from Git, a method was implemented to retrieve a full history of the repository. All of the instruction files relating to users for a given group are then parsed, creating a list of instructions for each user, indexed by timestamps.
The instructions for each user are combined, re-ordered and updated to start from time 0, and then parsed so each instruction is sent to the correct URL to re-create either a deployment or an injection on the Node-RED servers.



A method was implemented to allow the replaying of a subset of the users' performance so a specific contribution can be isolated, and furthermore, the performances can also be replayed on different IP addresses than the original ones.

User test

Music and specifically timing intolerances in music, and also the recording and replaying solution are all subjective, so a user test in a school was conducted.
The test involved having two bands of 5 students, each collaboratively creating music over the course of 1 hour; a questionnaire was used to gather data about the correctness of the solution.

Results: Beat synchronisation

The main question of interest was a ranking prompt of "the beat generation sounded synchronised", with answers ranging from 1 (strongly disagree) to 5 (strongly agree).

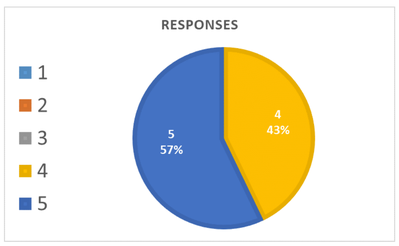


Figure: "Beat generation sounded synchronised"

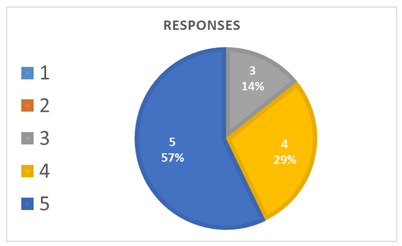


Figure: "It was simple to create music"

The first figure shows how the students thought the beat generation, including tempo changes, was synchronised. The second figure shows how the majority of the students found it simple to use this system for collaborative music generation, however a few students found it more complicated, and this is backed up by responses to the more open-ended questions, with one student identifying "setting up" as the worst thing about the session; observations from the test also highlighted how some steps of setting up the synchronisation were not so intuitive and easy to mess up.

Results: Recording and Replaying

The question used to evaluate the recording solution was "The replaying solution sounded the same as the initial performance", with answers ranging from 1 (strongly disagree) to 5 (strongly agree).

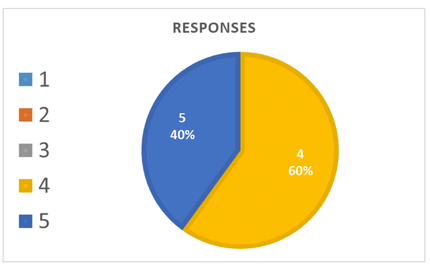


Figure: "The replaying solution sounded the same as the initial performance"

This figure shows how the participants thought the recording solution worked effectively to record the collaborative music performances generated.