We predict that it is possible to maintain a coherent musical collaboration through a wide area network if latency and jitter are below some threshold. We will test this hypothesis by emulating different network conditions using dummynet (or a similar tool). Using dummynet, we can control packet delay and jitter and investigate the effect they have on the coherence of a distributed music session, thus determining whether coherence is observed when conditions are under the threshold. We will measure coherence by comparing the sequences of OSC messages received by each SuperCollider instance -- looking at the events describing the sounds each user is about to play. If these sequences are fully identical, so that messages are in the correct order (before they are scheduled to be played), we will conclude that the session is in a coherent state. We will measure latency and packet loss rate in both the University network, and a home ISP to find out what kind of network conditions we are likely to encounter (on average) -- this will help guide the emulation of network conditions using dummynet, ensuring the system is able to deal with realistic scenarios. Using these findings, we will attempt to improve our implementation to automatically tune the scheduleAheadTime of the separate SonicPi instances to adapt to different network conditions -- keeping scheduleAheadTime as low as possible so users experience the best possible response time (in audible terms) when submitting changes to their own scripts.

Using SonicPi’s scheduleAheadTime parameter, we hope to react to the latency, jitter and skew problems described above. For any two users A and B, maintaining coherence for A is possible if its scheduleAheadTime is set to be more than or equal to the sum of the latency between A and B, and the skew in time between B and A within SonicPi (subtracting A’s notion of time from B’s notion of time). We will measure latency and packet loss rate in both the University network, and a home ISP. Using these findings, we will attempt to improve our implementation to automatically tune the scheduleAheadTime of the separate SonicPi instances to adapt to different network conditions -- keeping scheduleAheadTime as low as possible so users experience the best possible response time (in audible terms) when submitting changes to their own scripts.

How to set the threshold

We wish to calculate this threshold to establish under what conditions we are unable to provide a deterministic view of events for all users.

We aim to create different versions of the system using TCP and UDP and use the protocol that provides the best performance. We first need to determine how costly packet delay, jitter and packet loss are to the quality of the music session (in terms of coherence). We will measure how much adjusting these aspects individually (using dummynet) affects coherence -- for example, by examining the OSC messages received by all users in a session, and counting the number of messages that were observed in an incorrect order (or not received at all). Protocol performance will then be rated using measurements of the latency, jitter and packet loss rate that the network experiences when using the protocols (on average) -- by comparing the performance measurements to their relative costs on session quality, we will be able to establish which protocol provides the better results.