Network music and the algorithmic ensemble

Abstract

Network music happens when people make music with computer networks, and algorithmic approaches to network music introduce specific challenges and opportunities. Networking is an area of considerable complexity from a programming standpoint, involving the representation and handling of uncertainty and failure, and computer networks and networked forms are fundamental to contemporary governance and politics. The allure of network music lies both in this potential for play with key aspects of present-day power structures, and in its potential support for musical relationships of friendship, collaboration and participation. Key network music dynamics that emerge from the materiality of networking technologies revolve around considerations of latency and jitter, bandwidth, and security. Each of these dynamics is modified strongly when it becomes a matter not simply of network music, but more specifically algorithmic, network music.

by David Ogborn ogbornd@mcmaster.ca

This is a draft of a chapter/article that has been accepted for publication by Oxford University Press in the Oxford Handbook of Algorithmic Music, edited by Alex McLean and Roger Dean and published in 2018.

Introduction

The early twenty-first century is marked, among other things, by a proliferation of networking technologies. This proliferation is multi-dimensional: the sheer quantity of computer networking devices, their distribution over the space of the planet (and the space of everyday life in many but not all places), and the co-existence of different types of networking devices — from the venerable and robust Ethernet, to rapidly succeeding generations of wireless signals, to the networks within networks of virtual private networks (VPNs) and other darknets, to tentative gropings with decentralized mesh networks. This proliferation of networking technologies is no less visible, or rather audible, in music, than in any other field of human endeavour. The emerging field of network music brings together a wide range of musical experiments that take these readily available (and sometimes not so readily available) contemporary networking technologies and make music with them. Frequently, the design and deployment of algorithms goes hand in hand with this artistic activity.

Network music happens when people make music over, or through, a computer network. On account of the ubiquity of networking technologies in contemporary everyday life, it is perhaps useful to add a further qualification: network music happens when people make music that explicitly depends on the affordances or materiality of computer networking technologies. While it is possible to find earlier precedents for such activity, the field of network music has gained momentum throughout the first decades of the 21st century, as evidenced by new hardware and software systems, ensembles and research groups, and festivals, such as the annual Network Music Festival in Birmingham, United Kingdom or the TransX Transmission Art festival in Toronto, Canada. Network music takes a large and ever-growing number of distinct forms, and network music practices can be broadly characterized along two axes of remote vs. co-located and

synchronous vs. asynchronous collaboration (Barbosa 2003). A musician or ensemble might use network music techniques to project their performance in one location to other, remote locations. A group of musicians may play together although they are distributed geographically, whether to adjacent rooms in the same building or to opposite corners of the world (and everything in between). Network music performances may involve elaborate attempts to construct a sense of co-presence through immersive audio and video streams, while other network music performances will eschew such “realism” in favour of more “abstract” forms of musical cooperation. While network music most often involves performing together at the same time, situations where networked collaborators take turns over longer spans of time are also possible.

Networked, algorithmic music is at an exciting turning point in its development. The rise of an energetic live coding movement is happening in parallel with the arrival of more powerful audio and video sharing mechanisms, and the potential of the web audio API to make algorithmic music and audio languages widely accessible has only begun to be explored. Alongside regular network music appearances at festivals and conferences, ad hoc network music events are becoming increasingly common. The broad reach of social media and other modern conveniences of connection and communication allows such events to be organized from anywhere in the world and receive an audience. Indeed, one of the potentials of network music most generally is to reduce or remove impediments to wide attendance and participation in artistic events (Boyle 2009). **note to editor: we should crossref the live coding chapter**

Figure 1: *Members of the Cybernetic Orchestra, connected via Ethernet, at the 2013 live.code.festival in Karlsruhe, Germany (clockwise from top left: Amy McIntosh, Jason Rule, Heather Kirby, Kearon Roy Taylor, Aaron Hutchinson, Dima Matar, Elise Milani, Myles Herod and David Ogborn)*

This chapter begins with a closer examination of the allure of network music, and then continues with a

discussion of the nature and influence of key network music dynamics that are more or less bound to the materiality of the networking technologies: latency (and jitter), bandwidth and security. Each of these dynamics is modified strongly when it becomes a matter not only of network music, but also of algorithmic music.

Why do we play with networks?

Contemporary digital computers are themselves networks. A computer consists of a discrete set of elements (memory locations, registers, input and output transducers, etc), and each of these elements occupies a different point in physical space (for example, a different location on a motherboard, or within an integrated circuit). In response to machine language instructions, electronic signals pass between these points. High-level programming languages tend to conceal this small-scale networking, instead encouraging us to think of the computer as a unified organism (a so-called “black box”), to think of data as occupying no physical space, and to think of the materials of computation as immediately and absolutely present.

At larger scales, the act of communicating signals from one location to another becomes more clearly recognizable as a matter of networking. When the body of the computing device is pierced by cables of varying lengths, when the transmission from point A to point B becomes much less robust (and also an easier target for surveillance), when the probability of a desired response arriving in return falls — in all such cases, no one would doubt that networking is present. In any case, between what is imagined as internal to a computer and what is clearly networking lies a clear difference in terms of user access — I can choose to connect my Ethernet cable to whatever I like, and I can (hopefully) turn off my WiFi connection, but rare indeed is the artist who rewires the connections between their memory chips and their processor (one exception would be circuit-bending performer Jonathan Reus, who intervenes directly in the electronics of older iMac computers to listen to the signals found therein and to distort them).

High-level programming interfaces reinforce this distinction between the individual machine and the network. The assignment “ $x=x+7$ ” is a basic, trivial and reliable operation whose comprehension is considered an elementary matter of computing education. As soon as x becomes an entity “out there” on a network, the operation is no longer basic, trivial or reliable, and rarely a part of anyone’s first steps with writing algorithms.

There is a potential analogy between the way that most computer languages represent time and the way that they represent the network outside of the individual computing machine. For the most part, programming languages have only awkwardly and imprecisely represented times and durations, with the dominant assumption being that instructions execute “as fast as possible” rather than at some very specific time, which sometimes leads to difficulties for a domain like music wherein the temporal positioning of events is fundamental. In the same way, programming languages deal clumsily with the fact that our computers are almost always connected/networked to numerous other things that are themselves connected/networked to numerous other things, etc. Given the recent appearance of music and audio programming languages that represent time in precise ways, such as *ChucK* (Wang and Cook 2003), *Extempore*, and *Tidal* (A. McLean 2014), perhaps we can also hope for new programming language designs that represent networking no longer as the unreliable outside of the machine, but rather as the everyday reality that it is.

In application, networks are about space and its reconfiguration. In the simplest sense, that is what makes a network a network — that an intensity in one spatial location is somehow carried, more or less systematically, to another spatial location. Our contemporary networks create new forms of space, while previous forms of space are frequently remediated within new network spaces. Popular virtual worlds present a readily comprehensible example of this capacity of networking to create new spaces as folds

in between real-world spaces. A prominent network music ensemble, the Avatar Orchestra Metaverse, performs from “within” the Second Life virtual world, but of course their performances are always somehow projected into the multiple real-world spaces occupied by the performers and their audience.

Almost any deployment of screens and loudspeakers, sensors and input devices, creates a new space though, and ideas about space are intimately connected to ideas about governance and social dynamics. A new type of public space produces a new type of public sphere, and network music, in producing new variants of spaces for public revelation and action is in a very fundamental way an imagining, or reimagining, of how people can be connected to each other and live together (Baranski 2010). Networking, then, can be productively elided with musicking, understood by Christopher Small as the establishment of relationships that “model, or stand as metaphor for, ideal relationships as the participants in the performance imagine them to be: relationships between person and person, between individual and society, between humanity and the natural world and even perhaps the supernatural world.” (Small 1998, 13)

If networks are about relationships between people, then they are also about power, control, and governance. The everyday perception that our lives are evermore influenced by hidden algorithms is only possible because networks connect those algorithms to all of us. We might chart a progression from forms of governance based on a centralized authority, to those based on decentralized bureaucracy, to those based on distributed, networked, protocological forms, and any number of contemporary realities might be brought in to testify that this has not been a move from bondage into freedom, but rather a reconfiguration of the way things are decided, a reconfiguration of the way that we perform individually and collectively (Galloway 2004).

Whether consciously or unconsciously, and whether by reinforcing or resisting them, network musicians are playing with the tools and symbols of power in “our age”. Pioneering network ensemble “the Hub”, known at first as the League of Automatic Composers, explicitly rejected an ethos of control and determinacy in favour of surprise and complexity, connecting their machines to each other in decentralized feedback loops that would never produce the same sounding result twice (Bischoff, Gold, and Horton 1978). From the beginning of the 1980s, their system evolved to feature a central, hub computer or “Blob”, which acted as a panoptic shared parameter memory for all of the other computers, connected in via custom-built RS232 interfaces. Later, with the adoption of MIDI technologies towards the end of the 1980s, the shared memory was abandoned in favour of a practice of distributing and redistributing events targeted to specific receiving computers (Gresham-Lancaster 1998). One can construct a reasonable allegory of contemporary networked life armed with just these two alternatives: the panoptic database and the viral event.

As the example of the Hub suggests, the history of network music certainly does not begin with WiFi and the Internet. Nor does it begin with microcomputers and MIDI. If it begins anywhere (and the search for beginnings is always suspect except as a reminder of the many threads that are woven together in our present), it begins with radio and the telephone. While the complex history of radio’s proliferation would be out of scope here, we can at least underline that our contemporary networking technologies are descendents of a rich heritage of such technologies, and that, similarly, our network arts (our creative uses of these technologies) are similarly connected to longer traditions of radio art, transmission art, etc. By roughly the middle of the twentieth century, the radio – that diffuse yet universal networking technology – was largely used in a broadcast fashion, as a way of communicating out from the centres of power to the masses. As a reaction to this, traditions of radio and transmission art have often emphasized dialogic elements. In the 1980s, the Japanese radio artist Tetsuo Kogawa championed mini-FM stations and, more broadly, “polymorphous media” that do not make “molar groups” out of their listeners but rather encourage individual connections, based on self-controlled tools (Kogawa, n.d.). Later, Kogawa became known for encouraging people to build their own FM transmitters.

While radio art has become less common, in parallel with a decline of societal investment in broadcast radio, this dynamic continues to exist. Dreams of the Internet as a distributed utopia for the free

exchange of knowledge, culture and entertainment have by and large been replaced by a centralized flow of information to and from a limited number of centralized sites, offered by large corporations and almost invariably tied in one way or another to advertising, surveillance, or both. Art and music that presents an alternate, dialogical configuration for the network thus retains critical potential. Algorithmic network music, moreover, has an additional critical potential: to take a post-humanist approach that does not necessarily accept intimate conversation between two bona fide human entities as the principal measure of success. Algorithmic network music can critique and resist the heavily centralised networks known as ‘the cloud’ without substituting for them the romantic salon.

Nothing about the deep involvement of networks with contemporary forms of power and governance contradicts that when we make music with networks, we derive pleasure both from that musical building activity, as well as the element of social togetherness that it produces. Pauline Oliveros, a longstanding exponent of network music in its telematic form (making geographically distributed musicians co-present to each other), points to musical friendships as a primary rationale. In Oliveros’ words, “If you are on the East Coast and the musician you want to perform with is on the West Coast then there is a reason [to make network music].” In almost the same breath, she points to globalization as another reason (Oliveros 2009).

Indeed, our contemporary, centralized social media platforms use the element of playing together socially as their primary attractor (or rather distractor, as in distracting people from either thinking about the fine print of user agreements or objecting to the torrent of advertising content)! Network musicians play together with networking technologies, simultaneously deriving pleasure, cultivating social relationships and calling critical attention to the forms that permeate our everyday lives. The nature of the technologies that they use gives rise to a number of “perennial” network music issues, while the specific context of algorithmic music often introduces additional strategies and challenges that aren’t present when network music is aimed simply at making traditional, acoustic musicians telepresent to each other. The following sections review three key material dynamics of network music situations with an eye to the changes introduced by an algorithmic focus: latency and jitter, bandwidth, and security.

Latency and Jitter

It takes time for any signal to go from point A to point B through any medium. This delay is usually called *latency*. This latency can vary from moment to moment, for example as a consequence of a change in the route a signal travels, and such variation in latency is called *jitter*. Latency and jitter are basic “problems” encountered by all network musicians, with an effect on numerous musical aspects, including but not limited to synchronization, performer interaction, tempo and spatial imaging.

The diameter of the planet Earth is 12,742 kilometres, and the speed of light is 299,792,458 m/s. An electromagnetic signal sent straight through the body of the earth would thus arrive at the other side 42 milliseconds later. In practice, of course, network cabling does not bore straight through the centre of the earth nor take the shortest possible surface route, but rather runs along its surface in complex diagrams. It is easy to think of possible surface routes that sum to 10,000, 20,000 or more kilometers. When network music events are globally distributed, it’s easy for the sheer travel time at the speed of light to add up to a delay that is quite apparent to our perception. In short, at these scales the speed of light feels surprisingly slow.

Additionally, long and short network routes both introduce various stages of buffering — the signal goes from one hop to the next, with a small delay introduced at each hop. Networking hardware and software introduces additional layers of buffering and delay, as does the interpretation or rendering of the transmitted information by application-level software. All of these inevitable delays sum to produce a

minimum possible latency for a given network route. In practice, these “non-transmission” delays can often be orders of magnitude greater than that of the speed of light. For example, the round-trip time to send audio between Hamilton, Canada and Montréal, Canada in the tabla and live coding duo very long cat is around 111 milliseconds — about 29 times longer than the approximately 3.8 milliseconds it would take a direct signal at the speed of light to travel the 567 kilometres that separate the two Canadian cities (Ogborn and Mativetsky 2015). This leads to a first commandment for all network musicians: never rely on predictions of what network conditions “should be” — measure what they really are!

The impact of “non-transmission” delay (hardware and software buffering, etc) is particularly evident when network music performances are not globally distributed but localised. For example, recent years have seen the emergence of a number of laptop orchestras, more or less large ensembles of musicians with laptops, often spatially distributed across a single acoustic space (like the large ensembles of any number of historical traditions of music-making), with their individual machines connected by some form of local networking, such as WiFi (Tsabary 2014; Trueman et al. 2006). In these situations, the raw electromagnetic transmission time becomes insignificant, and yet with typical networking hardware and software significant delays will still be experienced. To give a common “challenging” scenario: with a poor quality WiFi connection and heavy encryption/security measures engaged, one can easily encounter delays around a quarter of second, even with everything in the same room. Simplifying or eliminating the network encryption will reduce the latency but at the cost of inviting security problems (see “Security” below). More elaborate responses involve algorithms to synchronize the clocks on separate machines, scheduling musical events relative to these synchronization algorithms and forming local caches of shared musical parameters (Ogborn 2014; Ogborn 2012; Sorensen 2010).

The inherent latency of network transmission is not a problem when such transmission is in one direction only, from a sender to a distant receiver, as in the case of broadcast radio or the case of a contemporary Internet consumer streaming a video from a centralized service. Indeed, in such cases the latency is literally imperceptible once the decoding and projection of the streamed transmission has begun. When events become dialogical, however, the latency becomes more perceptible, and when the multi-way exchange carries precisely synchronized musical events it becomes more perceptible still. The question thus arises: how much latency is acceptable for a musically satisfying situation?

One common approach to this question begins by comparing network latencies to the delay involved with the mechanical propagation of sound signals. At 20 degrees Celsius and sea level, sound travels at 344 metres per second. On this basis, a small network transmission latency of 5 milliseconds is equivalent to the time it takes sound to travel 1.72 metres (i.e. also 5 milliseconds). A 20 metre distance from a proscenium stage to the middle of the audience might be compared to a 58 millisecond network transmission latency. While such comparisons are helpful, it should not be overlooked that these two types of delay are, in common practice, not mutually exclusive but rather effects that sum to produce a new acoustic situation characterized by (among other things) increased direct wave delay time relative to either network transmission or acoustic propagation taken in isolation.

Another approach to the question of how much latency is acceptable bases itself on selected results of psychoacoustic research. For example, early research on auditory perception showed that a threshold of roughly 20 milliseconds between two distinct sound events was required for a listener to identify which precedes the other (Hirsh 1959). With high quality local network connections, or Internet connections over shorter distances, it is certainly possible to achieve network latencies below this threshold. This is not the full psychoacoustic story, however. Even when we cannot identify which of those two distinct sound events come first, we can recognize them as distinct from a single, fused sound event down to time differences of around 2 milliseconds (*ibid.*). This rather more stringent figure will be no surprise to connoisseurs of digital audio interfaces. The manufacturers of such interfaces aim to produce the lowest possible conversion latencies in order to support real-time monitoring of signals transformed by software, and are cognizant that quite small latencies are perceptible to musicians who are monitoring their own

signals (or transformations thereof) during recording sessions.

A more recent strand of research approaches the question of latency not from, or not only from, the standpoint of audience perception, but rather from the standpoint of its objectively measured impact on musical performance actions. In one recent study, with pairs of rhythmic clapping performers separated by calibrated delays between 3 and 78 milliseconds, the effect on ongoing musical tempo was measured. This revealed four distinct phenomena: below 10 ms, the performers tended to accelerate; between 10 and 21 milliseconds, their tempo was stable. Above this and up to 66 milliseconds of latency, they tended to decelerate, due to the readily comprehensible behaviour of waiting for a collaborator's delayed metre. Above 66 milliseconds synchronization deteriorated rapidly (C. Chafe, Cáceres, and Gurevich 2010).

Latency, in short, has musical effects whose characterization evades a simple threshold of “good enough” versus “not good enough”. As the fields of music cognition and music information retrieval advance, their analytical instruments may be used to build an even richer picture of the effect of latencies (acoustic and network) on musical performance. In the meantime, none of these results should be taken as an indication that a particular latency is simply “good enough” for all time. From a position of very low network latency, longer latencies (alongside other virtual acoustic features) can always be simulated. But the inverse does not hold true: once the signal has been delayed 300 milliseconds from arriving at a given point in a system, it will forever be that 300 milliseconds later. High latencies don't only have direct effects on musical perceptions and performance — they are also constraints on what can be simulated or modelled in musical systems.

The algorithmic, network music context introduces an additional reason to pay close attention to the phenomenon of network latency. The performance of algorithms in a network, distributed space, can give rise to situations where small discrepancies in when a given piece of code executes result in large discrepancies in what the result of such code is. One example of this would be the use of oscillators. If, for example, a live coding artist creates identical sine wave oscillators on distributed machines and then later executes a second piece of code on all those machines that accesses the output of that oscillator they could get wildly different results depending on the time that has elapsed between the two pieces of code running on each machine, as a consequence of jitter. If the frequency of that oscillator is 125 Hz, then a discrepancy of only 2 milliseconds (a quarter wavelength of a total wavelength of 8 milliseconds) is sufficient to make a difference between the maximum absolute values of the oscillator and the minimum. Jitter can easily reach these magnitudes on dedicated wireless networks, and on the general-purpose Internet jitter typically far exceeds these magnitudes.

At the same time, the algorithmic, network music context also provides additional strategies for reducing the perceptual and musical impact of network latency:

1. Given relatively synchronized clocks (i.e. a frame of reference), algorithms can be written in such a way as to make things happen at aligned times in the near future. In effect, this is the direct response to the above-mentioned example of the low frequency oscillator. If the multiple, distributed instances of a low frequency oscillator start at the same time (given a known frame of reference) then they will have a deterministic result at some later known time (given a known frame of reference). This is also the approach taken by the ninjam network music software, which takes compressed audio performances from a given node in a distributed ensemble and delays the monitoring of that performance at all other nodes to line up with a subsequent period in the music (for example: my performance in a given 4 bars is heard by my collaborators as precisely lined up with the next 4 bars of the music) (Ninjam, n.d.).
2. Alternately, since an algorithm already implies a temporal gap between its specification and its realization, this can be exploited to create the illusion of simultaneity. One can deliberately delay the local monitoring of the algorithmic result from a given node in the network to line up with the delayed reception of results from other nodes in the network. There is no limit to how many

live coding/algorithmic nodes can be aligned in this way, and it is also possible to include one live audio (i.e. non-algorithmic) performance in a network ensemble using this technique (Ogborn and Mativetsky 2015).

3. Events can be structured in such a way that they do not need to be aligned between nodes at different relative latencies (and are “immune” to jitter). This has been a common strategy in the general evolution of network music — to accept the inevitability of latency and jitter, incorporating them into the musical fabric, or otherwise adapt to it (Tanaka 2006; Juan-Pablo Cáceres and Renaud 2008). In its simplest form, this can entail avoiding firm metric structures, or including layered drones, textures or other relatively rhythmically independent elements so that the timing discrepancy from one node to the next does not become obvious. The algorithmic network music context provides an additional variation of this latency strategy: code can be structured in such a way that it can be rendered or realized completely independently on different machines. Provided that the code to be rendered does not refer to other time-varying functions (or random functions), the result can be identical at different locations even at drastic relative latencies.

An extreme example of using the inherent latency of network music to advantage is provided by the SoundWIRE technique, which uses the delay of network transmission as the delay component of a physical modeling synthesizer, typically a Karplus-Strong model of a plucked string (Karplus and Strong 1989). A system is configured such that audio is sent to a node and then returned to where it came from, and then monitored, filtered and recirculated on the network (after a low-pass filter). The monitored sound becomes a sonic representation of the underlying network conditions, whereby low latency will produce higher pitch, and low jitter (variation in latency) will produce minimal vibrato (C. Chafe and Leistikow 2001).

Bandwidth

Driven by an orientation towards telepresence, network music has frequently resorted to the transmission of high-quality digital audio and video signals, creating links between distinct spaces through the medium of video projections and the “sound screens” that exist in and between arrays of loudspeakers. In addition to latency and buffering issues, these techniques consume very large amounts of bandwidth.

For example, a single, raw mono bidirectional 44,100 Hz digital audio signal at 24 bits per sample will require an absolute minimum of 1 megabits per second (Mbps) in each direction (44,100 times 24 is 1,058,400). In practice, the requirement is greater as some amount of redundancy is required in order to make such a network audio signal robust to long drop outs. This is especially the case when it is a matter of audio transmission over the Internet.

The *jacktrip* software is in wide use by network musicians, as a free, open-source and readily available means to stream audio over networks. *jacktrip* transmits uncompressed audio (i.e. buffers of linear PCM samples) in order to avoid the additional latency that would be introduced by encoding/decoding operations (J.-P. Cáceres and Chafe 2009). To compensate for network problems, *jacktrip* sends redundant copies of the audio data. This tends to demand more, and more reliable, bandwidth than is commonly available in home situations. Results tend to be strongest with either local Ethernet networks, or very robust Internet connections such as are sometimes available in universities and other research institutions. A more recent *jacktrip* server can receive many client connections, mix them, and redistribute mixed signals back to the clients (J. Cáceres and Chafe 2010).

At the time of writing, extremely robust standards have been put in place for the distribution of lossless Audio over Ethernet (AoE), and devices implementing these standards are increasingly common in high-end institutional settings, such as theatres, concert halls, and recording studios. While there is a large market for these devices (which considerably simplify the physical running of cables in spaces), there is also a

confusing competition between both open and proprietary standards. Like *jacktrip*, the AoE formats generally transmit uncompressed, lossless audio and require reliable network connectivity, such as is rarely found on ‘consumer grade’ Internet connections. Indeed, the Audio Video Bridging (AVB) standard requires specialized network switches.

A number of formats exist for lossy streaming of audio signals, including the Constrained Energy Lapped Transform (CELT) and its successor, OPUS. These formats have the advantage of using significantly reduced bandwidth, and also of being open standards. OPUS is supported by the Internet Engineering Task Force (IETF) and has been developed to introduce a relatively small “algorithmic delay” (i.e. the component of the overall latency due to encoding and decoding of the audio signal). While these haven’t yet been widely used in the network music and algorithmic music communities, one can expect that experimental energy will travel down these paths as software integrating these formats with practical, working algorithmic and electronic music systems appears.

However, the algorithmic music context provides an alternative or addition to all three of the above, relatively bandwidth-hungry methods of supporting telepresence; code (algorithms in text form) can be transmitted as low-bandwidth text data, and then executed either immediately or (better) on some definite schedule against a synchronized clock. In the extramuros software, for example, any number of live coding performers collaborate on shared code that appears in a web browser interface. Like with the popular Google Docs word-processing platform, each performer’s changes to the code are visible to each other performer in real-time. When a performer triggers the evaluation of some piece of this shared code, it is transmitted to any number of “client” computers for rendering into sound by whichever programming language the ensemble chooses. The combination of a shared editing interface with any number of rendering computers allows the software to be used in diverse network music settings, from workshops where participants bring nothing but a web-browser and connect over a local area network, sharing a single local projector and sound system, to globally-distributed ensembles where each participant renders the audio independently at their own location (Ogborn et al. 2015).

Network music thus involves the transmission of two broad types of musical data, and a given network music performance might involve either type of data in isolation, or it might combine them into a hybrid network music topology. On the one hand, there is data representing continuous audio and video signals, and on the other hand, data representing discrete objects or events, including isolated musical parameters and perceptible notes and events, as well as more highly articulated objects like code structures. There is a long history of musical networking systems based on the transmission of discrete objects and events, from the earliest sequencers, through the standardization and widespread adoption of the MIDI protocol, to the comparatively recent spread of Open Sound Control (OSC) (Wright, Freed, and Momeni 2003). It is possible to see the recent trend towards the transmission and distribution of code as an extension and abstraction of these more longstanding networking practices.

Pieces built on these two broad types of musical data are found in the repertoire of the Birmingham Ensemble for Electroacoustic Research (BEER). In BEER’s *Pea Stew* continuous audio signals (sent with *jacktrip*) are recirculated around a network of performers, with FFT-based phase shift as well as other live-coded transformations applied at each node. In another BEER piece, *Telepathic*, the network is used to share parameters — establishing a shifting, centralized tempo as well as quantizing the redefinition of three different musical layers by individual live-coding ensemble members, so as to produce unified, drastic changes in the resulting texture (Wilson et al. 2014).

Beyond their bandwidth parsimony, systems based on discrete objects and events have the further advantage of there being natural ways to adapt and modify the realization of the transmitted data to reflect “local” conditions, which could include the presentation of code intentions on Braille devices, screen-readers or fantastic kinetic sculptures. Live coding, by projecting both the code and its result has always been a kind of projectional editing, where intentions are communicated or “projected” in multiple ways (Walkingshaw

Figure 2: *Members of the “shared buffer group” performing at ICMC 2015 via extramuros, from four different global locations (clockwise from top left: David Ogborn, Ian Jarvis, Alex McLean and Eldad Tsabary)*

and Ostermann 2014). There is a massive and exciting space for play around the development and exaggeration of these projectional possibilities. The transmission of algorithmic art as discrete events (rather than audio and video signals) increases the exposure of the art to differences in the execution context. While it is sometimes imagined that the code is equivalent to its realization, this translation is no mere algebraic operation and this can become quickly exposed when code is being simultaneously realized by machines in different places and conditions. Network art can thus provide practical explorations of the mystery and unpredictability of software that is pointed to by software studies (Chun 2011).

A simple demonstration of this is provided by the laptop ensemble context: take a group of computers each connected to their own loudspeaker and then trigger some synthesis events on each of them from a “central” computer over a network. The result will, at a minimum, demonstrate differences in timing due to the way each computer’s network stack works, as well as differences in timbre due to each computer’s audio subsystem. While an oscillator at a given frequency is mathematically well-defined, it can quickly take on surprising details and differences when rendered by a specific computers connected to specific speakers and ultimately, specific ears. Powerbooks Unplugged (J. Rohrer et al. 2007) were early explorers of this terrain, using WiFi and just the built-in speakers of their laptops and with a SuperCollider-based system allowing small pieces of code to be rendered as sound on any of the machines, or any combination of machines, in a spatially distributed ensemble with all of the sound coming out of the small built-in speakers of each laptop.

Security

If networks are, fundamentally, about power, governance and public space, it should be little surprise that they can be seen through the lens of security. Indeed, network music events routinely run into practical obstacles that can be directly traced to one or more contemporary information security issues. Despite this, comparatively little attention is paid to such issues in the development of network music software and environments. This is a research and creative space that is ready for considerable expansion.

When network algorithmic music is a matter of a closed group performing on a wired Ethernet that they control, the possibility of security problems being deliberately caused by an outside agent is relatively minimal. Even in such an environment, security issues frequently come up as the machines themselves will typically be used in other less secure settings, and will “import” issues from elsewhere into the closed situation of the group. In laptop orchestras, there are always firewalls — necessary for the student who frequently connects to the Internet in public cafés no doubt — that need to be turned off (or have exceptions added to them). There is occasionally the computer whose web browser has been damaged by malware. Closed, secured WiFi networks introduce additional latencies, while open, unsecured WiFi networks expose ensembles to mischievous bystanders who might interfere with a performance by sending “unauthorized” OSC packets to shared systems. It is even possible to crash some of the common audio programming applications with the right OSC message, if that message is not sanitized before subsequent processing (Hewitt and Harker 2012).

When the Internet is used as part of a network music event, practitioners frequently run into the firewall structures of home and institutional networks. In the most common default configuration, these firewalls are configured to reject incoming communications but accept (and continue) outgoing communications. This scenario is closely bound up with the re-centralization of the Web in “Web 2.0”. If the primary usage scenario for the network is home users “surfing” a limited and stable selection of large central content “providers” (i.e. distributors) then that default firewall model makes a lot of sense. However, it tends to work against individuals temporarily placing their own web servers on the Internet, and thus against ad hoc multi-nodal network music applications. While not impossible, but the grain of the network tends to work against people doing these things without another level of either willpower or experience with the configuration of networking devices.

When algorithmic music environments are deployed permanently on the open Internet as platforms, the security issues become even more significant. Moreover, the specific domain of networked algorithmic music may produce requirements that aren’t fully met by the dominant approaches to securing networked appliances (requiring logins or even more robust credentials). A public artwork involving a collaborative text editor on the Internet no doubt requires some protection against real-time spam, but passwords and credentials require coordination, and frequently become stumbling blocks to fluid collaboration, or at least stumbling blocks to beginning that collaboration. In the “real-world” musicians don’t require credentials in order to recognize that they share a space and not destroy the space that they are sharing. There is much potential for the design of spaces for networked collaboration around algorithmic music that take account of particular security requirements: we often want or need people to openly and freely participate and standard Internet security apparatus could impede that.

With the caveat that much more work does need to be done to make networked algorithmic music production environments robust to security issues (to allow the directors of laptop orchestras to get better sleep the night before a show, among other reasons), it would be a mistake to see information security issues simply from the standpoint of protecting oneself from threats. Indeed there is significant critical and creative potential in realizing works of art and music that experiment and explore and thus make representable and comprehensible security-related phenomena. We might listen to a distributed denial of service attack, spoof a SuperCollider server, or force a “shy” performer to share their screen via van

Eyk phreaking — giving new meaning to the venerable “show us your screens” (Ward et al. 2004)! We might imagine live coding battles where one part of an ensemble writes (and shares with the audience) algorithms to identify and counteract “malicious” code, while another part of the ensemble redoubles their efforts to write code that gets through the filters of the other part of the group.

Future Directions

Two interesting roads little traveled in network music are (a) the use of analyses derived from images and signals to defeat latency barriers, such as using visual information to predict when a percussionist will strike a drum ahead of the actual event, with a simulacrum of the sounding result synthesized at a destination site ahead of what latency would otherwise allow (Oda, Finkelstein, and Fiebrink 2013); and (b) the centralized rendering of algorithms connected to facilities that stream back the results, with the possibility of using enormous cloud-pooled processing resources to render things that could not be rendered by any domestic machine (Hindle 2015). The centralized rendering of sonic algorithms could be a promising road to providing accessibility to algorithmic music environments as presently many of them have significant installation challenges on the diverse machines and operating systems in contemporary circulation. A central rendering and streaming approach could remove the need for substantial installations in these situations — instead users would connect to a web-based editing environment, a server somewhere would render the result, which they would receive back via streaming.

The recently developed web audio API represents another way in which algorithmic music environments can be rendered immediately accessible to anyone with a web browser. With the web audio API, all the rendering of the audio takes place within web browsers, which have now become miniature, media-rich, standardized, quite secure operating system/virtual machines. Lich.js and Gibber are two recent projects that clearly display the potential of the web audio API to create zero-configuration entry routes into algorithmic music making, with both incorporating networked, collaborative editing (McKinney 2014; Roberts et al. 2015).

As part of the expansion of research around web audio, we can hope for the emergence of algorithmic network music platforms that facilitate the sharing of musical algorithms together with their results, as well as enable the formation of social communities around those tuples. Platforms have played an occasional role in the story of network music thus far, such as the asynchronous, primarily MIDI-based composition platform of netjam (Latta 1991), as well as Roger Mill’s use of the furtherfield.org platform to host a network music ensemble (Mills 2010). Nonetheless, it is a striking lacuna of the contemporary moment: we have algorithmic music languages, and we have maturing network music technologies, and we have platforms for sharing social connections and platforms for sharing media (i.e. freesound.org, soundcloud.com,) but little in the way of platforms for creating and sharing collective algorithm-sounds (things that are simultaneously algorithms and sounds/music). Building and experimenting with such platforms is an exciting avenue for future research, as such platforms would have the potential to reach a wide, international audience, while also blowing up worn but persistent ideas that the identity of the work consists only in the sound, or only in the sound-video composite, or only in the artist’s head/intentions, etc, i.e. only “in one place/sense” of some sort. Algorithmic art has always represented a healthy challenge to aesthetics that privilege sense intuitions — and new, networked platforms for the presentation of algorithmic art would be exciting, multiple, distributed places to be.

Bibliography

- Baranski, Sandrine. 2010. "Topographie de La Musique En Réseau Sur Internet." *Filigrane. Musique, Esthétique, Sciences, Société*. 12. <http://revues.mshparisnord.org/filigrane/index.php?id=295>.
- Barbosa, A. 2003. "Displaced Soundscapes: A Survey of Network Systems for Music and Sonic Art Creation." *Leonardo Music Journal* 13: 53–59.
- Bischoff, J., R. Gold, and J. Horton. 1978. "Music for an Interactive Network of Microcomputers." *Computer Music Journal* 2 (3): 24–29.
- Boyle, D. 2009. "Network Musics: Play, Engagement and the Democratization of Performance." *Contemporary Music Review* 28 (4).
- Cáceres, J.-P., and C. Chafe. 2009. "JackTrip: Under the Hood of an Engine for Network Audio." In *Proceedings of the International Computer Music Conference*, 509–12.
- Cáceres, J.P., and C. Chafe. 2010. "JackTrip/Soundwire Meets Server Farm." *Computer Music Journal* 34 (3): 29–34.
- Cáceres, Juan-Pablo, and Alain B. Renaud. 2008. "Playing the Network: The Use of Time Delays as Musical Devices." In *Proceedings of International Computer Music Conference*, 244–50. <http://classes.berklee.edu/mbierylo/ICMC08/defevent/papers/cr1425.pdf>.
- Chafe, Chris, and Randal Leistikow. 2001. "Levels of Temporal Resolution in Sonification of Network Performance." In *Proceedings of the 2001 International Conference on Auditory Display*.
- Chafe, Chris, Juan-Pabla Cáceres, and Michael Gurevich. 2010. "Effect of Temporal Separation on Synchronization in Rhythmic Performance." *Perception* 39: 982–92.
- Chun, Wendy Hui Kyong. 2011. *Programmed Visions: Software and Memory*. Cambridge: MIT Press.
- Galloway, A.R. 2004. *Protocol: How Control Exists After Decentralization*. Cambridge: MIT Press.
- Gresham-Lancaster, S. 1998. "The Aesthetics and History of the Hub: The Effects of Changing Technology on Network Computer Music." *Leonardo Music Journal* 8: 39–44.
- Hewitt, S., and A. Harker. 2012. "Security in Network Connected Performance Environments." In *Proceedings of the International Computer Music Conference*, 320–24.
- Hindle, Abram. 2015. "Orchestrating Your Cloud-Orchestra." In *Proceedings of the Conference on New Interfaces for Musical Expression*. <https://nime2015.lsu.edu/proceedings/244/0244-paper.pdf>.
- Hirsh, I. 1959. "Auditory Perception of Temporal Order." *Journal of the Acoustical Society of America* 31 (6): 759–67.
- Karplus, K., and A. Strong. 1989. "Digital Synthesis of Plucked-String and Drum Timbres." *Computer Music Journal* 7 (2).
- Kogawa, Tetsuo. n.d. "From Mini-Fm to Polymorphous Radio." *Subsol*. http://subsol.c3.hu/subsol_2/contributors0/kogawatext.html.
- Latta, C. 1991. "Notes from the Netjam Project." *Leonardo Music Journal* 1: 103–5.
- McKinney, Chad. 2014. "Quick Live Coding Collaboration in the Web Browser." In *Proceedings of the Conference on New Interfaces for Musical Expression*. http://nime2014.org/proceedings/papers/519_paper.pdf.
- McLean, Alex. 2014. "Making Programming Languages to Dance to: Live Coding with Tidal." In

Proceedings of the 2nd Acm Sigplan International Workshop on Functional Art, Music, Modelling and Design.

Mills, Roger. 2010. “Dislocated Sound: A Survey of Improvisation in Networked Audio.” In *Proceedings of the New Interfaces for Musical Expression Conference*, 186–91.

Ninjam. n.d. <http://www.ninjam.com>.

Oda, Reid, Adam Finkelstein, and Rebecca Fiebrink. 2013. “Towards Note-Level Prediction for Networked Music Performance.” In *Proceedings of the Conference on New Interfaces for Musical Expression*. http://www.cs.princeton.edu/~fiebrink/publications/OdaFinkelsteinFiebrink_NIME2013.pdf.

Ogborn, David. 2012. “EspGrid: A Protocol for Participatory Electronic Ensemble Performance.” In *Audio Engineering Society Convention 133*. Audio Engineering Society. <http://www.aes.org/e-lib/browse.cfm?elib=16625>.

———. 2014. “Live Coding in a Scalable, Participatory Laptop Orchestra.” *Computer Music Journal* 38 (1): 17–30.

Ogborn, David, and Shawn Mativetsky. 2015. “Very Long Cat: Zero-Latency Network Music with Live Coding.” In *Proceedings of the First International Conference on Live Coding*, 159–62. ICSRiM, University of Leeds. doi:10.5281/zenodo.19348.

Ogborn, David, Eldad Tsabary, Ian Jarvis, Alexandra Cárdenas, and Alex McLean. 2015. “Extramuros: Making Music in a Browser-Based, Language-Neutral Collaborative Live Coding Environment.” In *Proceedings of the First International Conference on Live Coding*, 163–69. doi:10.5281/zenodo.19349.

Oliveros, Pauline. 2009. “From Telephone to High Speed Internet: A Brief History of My Tele-Musical Performances.” *Leonardo Music Journal (Online Supplement)* 19.

Roberts, Charlie, Karl Yerkes, Danny Bazo, Matthew Wright, and JoAnn Kuchera-Morin. 2015. “Sharing Time and Code in a Browser-Based Live Coding Environment.” In *Proceedings of the First International Conference on Live Coding*, 179–85. ICSRiM, University of Leeds. doi:10.5281/zenodo.19351.

Rohrhuber, Julian, Alberto de Campo, Renate Wieser, Jan-Kees van Kampen, Echo Ho, and Hölzl Hannes. 2007. “Purloined Letters and Distributed Persons.” In *Music in a Global Village: Symposium*. http://www.wertlos.org/~rohrhuber/articles/Purloined_Letters_and_Distributed_Persons.pdf.

Small, Christopher. 1998. *Musicking: The Meanings of Performing and Listening*. Hanover: University Press of New England.

Sorensen, Andrew. 2010. “A Distributed Memory for Networked Livecoding Performance.” In *Proceedings of the International Computer Music Conference*. <http://eprints.qut.edu.au/55715/1/icmc2010.pdf>.

Tanaka, Atau. 2006. “Interaction, Experience, and the Future of Music.” *Computer Supported Cooperative Work* 35: 267–88.

Trueman, Daniel, Perry Cook, Scott Smallwood, and Ge Wang. 2006. “PLOrk: The Princeton Laptop Orchestra, Year 1.” In *Proceedings of the International Computer Music Conference*, 443–50. http://www.scott-smallwood.com/pdf/plork_icmc2006.pdf.

Tsabary, Eldad. 2014. “Music Education Through Innovation: The Concordia Laptop Orchestra as a Model for Transformational Education.” In *8th International Technology, Education and Development Conference (Inted)*, 657–64.

Walkingshaw, Eric, and Klaus Ostermann. 2014. “Projectional Editing of Variational Software.” In *Proceedings of the 2014 International Conference on Generative Programming: Concepts and Experiences*,

29–38. ACM. <http://web.engr.oregonstate.edu/~walkiner/papers/gpce14-projectional-editing.pdf>.

Wang, Ge, and Perry R. Cook. 2003. “ChucK: A Concurrent, on-the-Fly, Audio Programming Language.” In *Proceedings of the International Computer Music Conference*, 219–26.

Ward, A., J. Rohrhuber, F. Olofsson, A. McLean, D. Griffiths, N. Collins, and A. Alexander. 2004. “Live Algorithm Programming and a Temporary Organisation for Its Promotion.” In *Read_me – Software Art and Cultures*, edited by O. Goriunova and A. Shulgin. Aarhus: Aarhus University Press.

Wilson, Scott, Norah Lorway, Rosalyn Coull, Konstantinos Vasilakos, and Tim Moyers. 2014. “Free as in BEER: Some Explorations into Structured Improvisation Using Networked Live-Coding Systems.” *Computer Music Journal* 38 (1): 54–64. doi:10.1162/COMJ_a_00229.

Wright, Matthew, Adrian Freed, and Ali Momeni. 2003. “OpenSound Control: State of the Art 2003.” In *Proceedings of the Conference on New Interfaces for Musical Expression*, 153–59. http://www.music.mcgill.ca/musictech/nime/onlineproceedings/Papers/NIME03_Wright.pdf.