INCLUSIVE INTERCONNECTIONS: TOWARDS OPEN-ENDED PARAMETER-SHARING FOR LAPTOP ENSEMBLE

GRAHAM BOOTH

A thesis submitted to the University of Huddersfield
in partial fulfillment of the requirements for
the degree of Masters of Arts by Research

The University of Huddersfield
December 2010

Copyright Statement

Abstract

This research project concerns the use of networks in laptop performance, which allow players to directly shape the musical voices of their peers. The author's recent work with the Huddersfield Experimental Laptop Orchestra has highlighted a need to develop player-centred networks, in order to foster a wider exploration of interdependent approaches to musical performance. The core of this thesis details the design of a parameter-sharing system which aims to support a diverse set of approaches to performance, whilst allowing interdependencies to be flexibly reconfigured by players. The resulting system has been tested and evaluated with a cross-section of experienced laptop performers, with initial results showing that the system enables players to bring their existing experience to interdependent performance. In addition, the use of a high-level graphical model for manipulating interconnections allows a range of interdependencies to be explored at performer-level. Future work aims to support a broader range of performance practice and make the process of manipulating interconnections more intuitive.

Acknowledgements

With thanks to Professor Michael Clarke for providing invaluable feedback and guidance throughout. Special thanks also to Sam Birkhead, Julian Brooks, Sam Freeman, Scott Hewitt, Adam Jansch and Scott McLaughlin for their time and patience during the development and testing of this project.

Table of Contents

List of Figures

List of Tables

Word Count

20,925 words.

1. Towards Inclusive and Mutable Musical Networks

The key themes of this thesis are inclusivity and mutability as they relate to laptop performance. Within this opening section, I aim to trace a path through my own creative practice and that of others, in pursuit of an alternative method for sharing musical information, which is able to support a diverse range of possible approaches to computer-enabled performance. In Section 1.1, I begin by introducing the idea of the Interdependent Music Network and attempt to situate this approach with reference to other models of networked music practice, so as to highlight the potential of interdependent interactions to open up new areas for computer-enabled performance. In Section 1.2, I draw on my recent experience of working with the Huddersfield Experimental Laptop Orchestra (H.E.L.O.) to illustrate the practical difficulties of establishing a diverse yet interconnected ensemble. Here, the core themes of inclusivity and mutability are identified as they relate to a) the potential to accommodate a range of different approaches to performance and b) the ability to manipulate the underlying model of interconnection. Section 1.3 presents an overview of existing synchronous parameter-sharing systems and investigates their suitability for supporting each of these themes. The results of this survey inform a set of design aims for the Inclusive Interconnections system, which are summarised in Section 1.4.

1.1 Interdependent Musical Networks: From the Ground Up

Over the course of the last decade, the use of networks in music has opened up a rich area of practice, prompting further classification within the academic community (Barbosa 2003, Kim-Boyle 2008, Weinberg 2005). Such categorisations help to highlight the differing concerns of researchers and provide a context within which to situate new work. As part of one such study, Barbosa considers four categories of networked systems, comprising Local Interconnected Musical Networks, Shared Sonic Environments, Music Composition Support Systems and Remote Music Performance Systems. These cut across a broad range of recent concerns, many of which attempt to deal with the implications of geographically displaced performance over the internet. It is only the first, however, which considers interaction solely in the context of co-located performers and it is this area which will be the focus of the work described here.

Barbosa further defines Local Interconnected Musical Networks as *"groups of performers who interact in real time with a set of musical instruments (or virtual musical instruments)*

with sonic interdependency provided by a local computer network" (2003). Ivica Bukvic, director of Virginia Tech's Linux-based laptop orchestra L2Ork, describes the overall aesthetic of such interdependencies as follows: *"Imagine an ensemble where action of one performer, in addition to generating an aural event also alters properties of an instrument commanded by another performer. This kind of connectedness among performers could be used to produce a complex web of interdependencies, effectively rendering the entire ensemble as one huge meta-instrument."* (Bukvic 2009).

Despite the recent interest in this area, there has been significant prior exploration of an interconnected aesthetic in the work of groups such as the League of Automatic Music Composers. From the late 1970's onwards, the group explored the idea of interdependency by establishing parametric connections between players, thus enabling direct manipulation of each other's explicit musical voices during concert performance (Brown & Bischoff 2002).

In recent years the stage has been set for wider engagement with musical interdependencies, in line with the view that new work should *"reintroduce casual social contexts for making music"* (Gurevich 2006). This has been made possible by the proliferation of high-speed communications technologies and the inclusion of both wired and wireless network hardware as standard on laptops. In addition, there has also been an increased understanding of how these technologies can be used in an artistic context (Networked Music Review 2007). Recent approaches to music making in this context have explored areas such as the development of toolkits for musical parameter-sharing between improvising laptop performers (Burns and Surges 2008), methods of conducting players over wireless networks (Smallwood et al. 2008) and the creation of complex textural material in real-time from the gestures of individual laptop performers (Harker et al. 2008). Such explorations can be said to challenge and reframe existing notions of live performance in a way which is native to a networked approach. In this regard, the author agrees with the sentiments of Weinberg (2005), who views the central innovative concept of computer-based performance as *"the level of interconnectivity among players and the role of the computer in enhancing the interdependent social relations".*

So far however, the use of local networks in performance has largely been restricted to what can be termed closed situations. These are characterised by a top-down approach influenced by the uniformity and parallelisation commonly found in software design, where designers narrowly constrain the terms of interaction, often using a single fixed model of interconnection and an identical sonic palette for each player. This allows players to

participate in a nominal way, without actually being able to bring enough of themselves to the performance to make a distinctive contribution. In contrast, a bottom-up approach to establishing Interdependent Musical Networks can be envisaged which would a) allow interdependent connections to emerge and develop as part of the process of performance itself and b) take into account players' existing laptop performance practice. With careful design, such an approach has the potential to draw on the existing skills and approaches of experienced laptop performers, and in doing so encourage a wider exploration of the aesthetics of interdependence.

1.2 Harnessing Diversity in the Huddersfield Experimental Laptop Orchestra

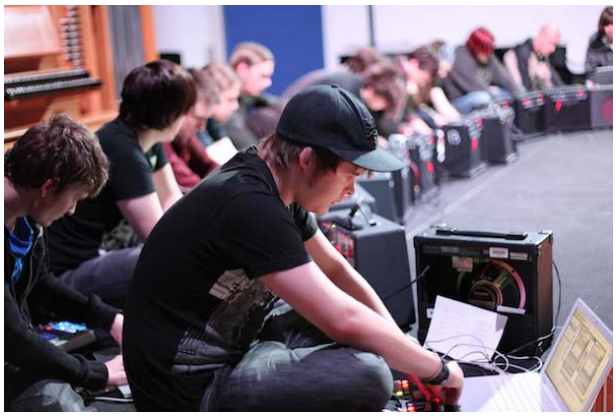


Figure 1: The Huddersfield Experimental Laptop Orchestra in performance

The Huddersfield Experimental Laptop Orchestra (H.E.L.O.) was founded in 2008 by Scott Hewitt as part of his ongoing doctoral research at the University of Huddersfield entitled *The Laptop as an Ensemble Instrument: Methods and Concerns* (Hewitt 2010). The ensemble performs both as a large-scale undergraduate group and a smaller postgraduate unit[1], with the former also acting as a valuable pedagogical tool[2]. In this respect, the undergraduate orchestra shares some common aims with other ensembles based in academic environments, such as the Princeton and Stanford Laptop Orchestras, whose approaches have been widely documented (Smallwood et al. 2008, Wang et al. 2008). However, in contrast to the approach of these groups, the H.E.L.O. undergraduate group has sought to

[1] H.E.L.O. has performed regularly at various national festivals and events, including the Huddersfield Contemporary Musical Festival (2009), Sonic Arts Expo (Leeds 2009) and FutureEverything digital arts festival (Manchester 2010).

[2] The ensemble is run as an assessed course module which is available to both Music and Music Technology undergraduates.

make a case for diversity in terms of the use of the laptop as an ensemble instrument (Hewitt et al. 2010).

1.2.1 Undergraduate Ensemble Practice

Within the undergraduate group, the aim is for each student to develop an individual approach to computer-based performance and to be able to demonstrate competence in a variety of concert situations. Rather than offering a definitive answer to the question of how to use the laptop as an ensemble instrument, the ensemble directorship instead encourages diversity in terms of the techniques employed by each participant. This is both for financial and ideological reasons. In the first case, members cannot be expected to purchase specific equipment or software to suit the wider aims of the group, whilst from a pedagogical perspective it is seen as an important part of developing a personal approach for participants to be able to draw on tools they are already familiar and to exploit these to develop new performance skills.

H.E.L.O. undergraduate group practice can be summarised as being a) inclusive, b) incubatory and c) assessed. With regards to a) and b), members may join at any technical or musical level, and are encouraged through weekly workshop sessions to adapt and extend their existing compositional or studio-based practice to deal with the challenges of real-time performance. Whilst in the past some members[3] have developed their own approaches using Max/MSP[4], the majority have drawn on their experience of commercial sequencing packages such as Live, Reason and Logic. Whilst these environments may not always be ideally suited to developing instruments which afford in-the-moment control, they are attractive in that they provide an initial point of access for newcomers, with an accompanying set of skills which can be reapplied to the performance domain. Over time players are encouraged to develop additional techniques and acquire new skills in response to the varying challenges of ensemble performance.

In terms of c), players participate in at least two assessed public concerts in the course of the academic year. In contrast to the freedoms outlined in the previous section, players are solely responsible for their own setup and have a duty to ensure their ability to perform both at rehearsal and in concert. Assessment of the undergraduate ensemble is conducted under real-world concert conditions, which have varied from reverberant church halls

[3] Typically those taking Interactive Sound Design as part of their programme of study.

[4] http://www.cycling74.com

(typically suited to acoustic chamber music), through to small club or stage environments (where space and setup time is often limited). With this in mind, the orchestra aims to adjust its approach and repertoire to suit the particular performance opportunity at hand.

As part of the assessment process, it is required by all final year undergraduates to undertake a leadership role of some kind. This may include leading or conducting the ensemble, developing a software performance tool, contributing to discussions on how the group should operate, or composing a piece to add to the concert repertoire. The majority of concert works are therefore drawn from the members of the group themselves, and historically these have explored both score-based and software-based approaches.

1.2.2 Postgraduate Ensemble Practice

The postgraduate group (H.E.L.O.pg) can be seen as extension of the above practices, but with some key differences. For example, members have typically developed more highly individualised approaches, which stem from greater experience in both instrument design and performance. In addition, there is generally less reliance on out-of-the-box software packages and a greater tendency towards self-designed software instruments or impromptu live coded methods. There is no unified approach to either hardware or software, with current members using the Renoise[5], SuperCollider[6], Max/MSP, MaxForLive[7], and ChuCK[8] environments. Hardware is similarly diverse, with some members using controllers such as the Monome and Novation Nocturn, whilst others rely solely on the native capabilities of the laptop itself. The group is also typified by being smaller in size than the undergraduate ensemble, with greater familiarity between members. Perhaps because of this, H.E.L.O.pg practice is almost always improvisatory in nature.

1.2.3 Approaches to Sound Reinforcement

A variety of approaches to sound reinforcement have been explored by both the undergraduate and postgraduate groups, with the most common of these being the use of individual guitar combo amplifiers. These quickly become an established part of each member's setup and whilst limited in dynamic range, they have proved to be an effective

[5] http://www.renoise.com/

[6] http://www.audiosynth.com/

[7] http://www.ableton.com/maxforlive

[8] http://chuck.cs.princeton.edu/

practical solution in most cases, due to the ease with which they can be transported to concert venues and the fact that they satisfy the need for both local monitoring and forward projection.

Other methods have been employed by the two groups as required. For example, inbuilt laptop speakers have been used by the undergraduate group on a number of occasions, most commonly where their characteristics have been treated as a particular compositional challenge. An example of the latter is the author's piece Faultlines, which takes the form of a constrained networked environment and allows participants to sound tones on each other's laptops (see Figure 2)[9] and embraces the freedom of movement provided when sound sources are not tethered to a power source (Booth 2010).

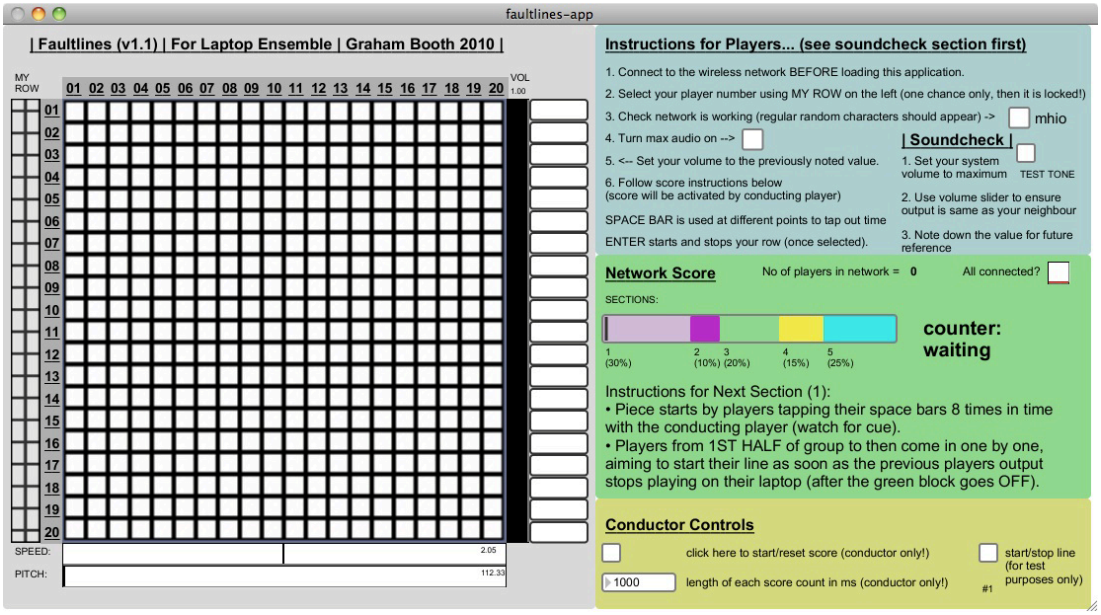


Figure 2: Graphical User Interface of Faultlines, a top-down networked composition for laptop ensemble.

In general, it can be said that the use of laptop speakers is more desirable in a larger ensemble or more intimate venue, where it remains possible to create a denser, more convincing sound. In contrast, the postgraduate ensemble regularly rehearses and performs using a stereo PA system. As there are relatively few members compared to the undergraduate group, each player's contribution proves easier to identify in the mix and the overall approach allows the ensemble to produce a detailed collective sound.

[9] Further materials for the piece can be found online at http://helo.ablelemon.co.uk/doku.php/materials/faultlines

1.2.4 Performance Models

In this section, I draw on a number of traditional instrumental performance models, as defined by Winkler (1998), in order to illustrate how some aspects of a performance may be pre-determined by a score, whilst others may be defined in-the-moment by performers. These examples will then be used to frame the practice described in the previous section, paying attention to the balance between pre-composed and improvised approaches in H.E.L.O. practice.

The first approach proposed is termed the *conductor model* and takes the symphony orchestra as a key example. Here, the conductor communicates aspects of a score (such as tempo and dynamics) to the orchestra in the form of physical gestures. The role of individual players is to interpret these high-level instructions as well as to realise lower level details which may or may not be encoded within the score. In contrast, the *chamber music model* is based on the musical interactions commonly found in a string quartet. Whilst the overall musical content is similarly governed by a score, this approach is characterised by a situation of reciprocal influence rather than one of top-down command. Here, players share control of aspects such as intonation, phrasing and tempo and it is this push/pull relationship which defines the creative space in which the composition is brought to life.

The next two models can be defined by their decreasing reliance on the idea of a written score. In the *improvisation model*, the practice of a jazz combo provides an insight into a form where traditional compositions ('standards') provide a common structure, which is then open to various level of interpretation. This ranges from the personal expression of solo parts, down to variation on the basic harmonic structure of the piece. This is only made possible by a high level of musical intelligence, based on shared experience. The logical extreme of this approach is the *free improvisation* model, which is defined by the complete absence of written material. Instead, both the overall structure and the details of the music are spontaneously formed in-the-moment by performers. This process is highly interactive and *"gives the performer much wider freedom to act, since he or she can simultaneously play the roles of interpreter and composer to create a dialogue with other musicians."* (Winkler 1998).

In terms of the different approaches taken by H.E.L.O., the *chamber music* and *free improvisation* models can be said to mark out key areas of practice, with the former describing the use of pre-composed work in the undergraduate ensemble and the latter encompassing the use of improvisation in both ensembles. It can also be said that when

scores are used, they are rarely conducted in a way that a symphony orchestra might be. This absence of the *conductor model* in H.E.L.O. practice can be attributed in part to a lack of shared instrumental knowledge within the group. For example, while conducting a composed work for a symphony orchestra can be said to draw on common knowledge of player's instruments and the type of interactions they afford, in the case of H.E.L.O. it can be argued that such a body of knowledge does not exist. This is due to the diversity of approaches employed within the group to develop instruments, combined with the lack of physical constraints placed on sound production. In comparison to traditional instruments, there is little tacit knowledge of what can be expected of a laptop instrument, and so this must be established in rehearsal and relies far more on active listening than on any shared implicit understanding. Therefore, when scores are used, they are often followed or interpreted in a similar way to the *chamber music model*, with reference to the surrounding sonic environment.

Free improvisation remains a mainstay of group practice for the similar reasons, which can be defined as the need to 'feel out' the nature of each player's approach, in order to begin to establish a meaningful exchange. Parallels with the more structured *improvisation model* may be also found when the idea of musical exploration is bounded to a particular soundworld, or structure. As with the jazz combo, this draws on the shared knowledge within the group about the nature of the musical material or form, but relies less on a clearly predefined body of performative knowledge.

1.2.5 Reconciling Diversity and Interconnection

As we have seen in the previous sections, both the undergraduate and postgraduate ensembles are characterised by a wide range of technical and creative approaches to performance that represent a cross section of current laptop performance practice. So far, however, it has been difficult to reconcile this diversity in an interdependent situation, due to the complexity of supporting these different approaches whilst also keeping the process manageable for players (who typically have enough to deal with in terms of ensuring the performance readiness of their own hardware and software). Instead, these concerns have tended to push interdependent approaches to performance in the direction of closed software applications, which place limitations on players' individual approaches to sound creation and the type of interactions possible.

It has been noted both within the author's own practice in Faultlines and within that of other laptop ensembles (Harker et al. 2008) that there is often an inversely proportional relationship between the diversity of the ensemble and its level of interconnection. Often, practice falls squarely into one of two camps: either a *diverse disconnected* approach, where players interact through active listening and/or visual cues, or a *uniform interconnected* approach, where custom-built software is used in an almost identical way by all members of the group in pursuit of a particular goal or aesthetic, which we have described previously as a 'hive-mentality' (Hewitt et al. 2010).

Whereas a *diverse disconnected* approach can be said to share a number of similarities with existing instrumental practice, a *uniform interconnected* approach requires players to interact with each other as an affordance of the user interface, which has more in common with multimodal forms of interaction such as those found in computer gaming. Whilst a closed software application may prove beneficial to a composer in constraining interaction within a particular expressive space (Candy 2005) or in encouraging players to focus on a particular creative goal, the uniformity of this method also discards much of the individuality present within the group.

For a laptop performer with his or her own individually tailored approach, this imposition of a *uniform interconnected* approach can be considered analogous to asking members of a symphony orchestra to switch to playing toy pianos to meet the aims of a specific piece. When applied uniformly across a laptop ensemble, not only does this method require players to rethink their approach to performance, it also smoothes out individual approaches to both interface design and sound production. The result is a method which is likely to suit the skill sets of some players and not others, requires a lowest-common denominator interface which is supported by all laptop players (i.e. trackpad and keyboard) and typically replaces diverse voices with generic ones. Therefore, whilst this may provide players and audience with a novel experience in the context of an individual composition, it fails to take into account the differing skills of individual group members. In order to address these concerns, a *diverse interconnected* approach is proposed, which aims to retain the diversity of practice found in a disconnected ensemble by supporting a wide-range of approaches to interdependent performance.

1.3 A Survey of Synchronous Parameter-Sharing Systems

Although a number of distributed applications have been developed which facilitate the sharing of musical parameters over a local network, many can be said to fall into the category of closed-compositional environments rather than open-ended systems (Gurevich 2005, Pazel 2000, Weinberg 2002). Within this section, I have chosen to focus on a subset of systems which have been designed with a degree of openness in mind, in terms of being able to devolve performance decisions down to individual players. These systems will be evaluated in terms their ability to a) integrate with a variety of approaches to laptop performance (termed *inclusivity*) and b) allow a number of models of interconnection to be explored (termed *mutability*). The infrastructure and topology used in each will also be commented on briefly. The systems and their network features are contrasted in Table 1.

For the purposes of this survey, a musical parameter is defined as a continuous stream of information which shares a common meaning for both the sender and receiver. Although other models of exchanging musical data exist, such as asynchronous methods which are suited to exchanging pattern-based or rhythmic data (Weinberg 2002), I have opted to limit the scope of this survey to synchronous systems, as prompted by the need to share musical information moment-to-moment in a group setting. One of the benefits of the real-time parametric model is that the concept is already well established and understood within the computer-music community. Parametric control came to particular prominence during the rise of the MIDI protocol, where continuous controller messages (CC) were employed to standardise communication between devices. This approach continues to persist within modern sequencer environments, and has been extended more widely by human-readable protocols such as Open Sound Control[10].

Name of System	Network Transport	Communications Protocol	Network Infrastructure
M.P.G. Carepackage	Unicast	Max/MSP messages	Decentralised
N.R.C.I.	Multicast	OSC	Decentralised
Bridges	Unicast	OSC/MIDI/TCP	Decentralised
D.O.T.	Unicast	OSC	Decentralised

Table 1: Comparison of parameter-sharing system network features

[10] http://opensoundcontrol.org/

1.3.1 M.P.G. Carepackage

The M.P.G. Carepackage[11] has been developed by Nathan Wolek in order to co-ordinate network communications between members of the Mobile Performance Group (M.P.G.) at Stetson University's Digital Arts program (Wolek 2010). The aim of the system is to allow players to exchange explicit types of control data in improvised situations. The package is built in Max/MSP and consists of two patches, *musiclinks* and *riddumbank*. The former handles initial network connectivity, as well as pitch and tempo control, whilst the latter allows rhythmic sequences based on a shared tempo to be synchronised across the group.

In terms of network infrastructure, a decentralised approach has been taken, with the *musiclinks* patch containing the necessary code to establish individual communication channels between players. This provides a useful initial model, which leverages the inbuilt multicasting features of Max/MSP[12] to build up an initial picture of the group, whilst allowing subsequent communications to take place directly between peers via UDP.

Regarding the network topology, players may communicate in either master or slave mode, where a master player attempts to impose their settings on the rest of the group, whilst slave players capitulate with this request. As it is generally undesirable to have more than one master on the network at any one time, negotiation must be used to decide who will cede control to whom. This need for extra-musical communication is addressed via an inbuilt chat protocol.

In evaluation, it can said that the M.P.G. Carepackage proves successful at providing a simple to use and easily modifiable starting point for creating interconnections which meets the needs of a specific ensemble. However, a number of issues would need to be addressed to make the system more inclusive and provide greater control over the model of interconnection. In terms of inclusivity, the M.P.G. Carepackage currently excludes non-Max/MSP users from participation, as the patches provided are intended for additional modification within the Max environment by end-users. This points to the need for an additional infrastructure in an open system, in order to facilitate external communication between a player's software environment of choice and a network-aware mediating application. This could be achieved using a standardised protocol such as OSC or MIDI.

[11] http://www.lowkeydigitalstudio.com/2010/04/mpg-carepackage/

[12] As provided by the java-based *mxj net.maxhole* object.

In more creative terms, players' ability to bring their existing performance practice to a networked scenario is impeded by the choice in the Carepackage to share pre-defined parameters such as pitch and rhythmic timing. Although it is intended that these parameters undergo further modification by players, the group is not free to define their own starting point in terms of the type of musical information they might wish to share. Whilst this no doubt simplifies operation for users and *"unifies the musical character of the interaction"* (Wolek 2010), it also mitigates against novel use. Finally, in terms of mutability, the simplified master/slave architecture employed restricts the range of interconnection topologies that can be explored. For example, one-to-many mappings are possible, but not one-to-one or many-to-one (Rovan et al. 1997).

1.3.2 Network Tools for Collaborative Improvisation

Network Tools for Collaborative Improvisation[13] (N.R.C.I.) is a suite of tools written for the PureData platform[14] by Greg Surges and Christopher Burns of the University of Wisconsin-Milwaukee. The purpose of the toolkit is to enable members of the Milwaukee Laptop Orchestra to quickly combine synthesis and control modules for use in improvised performance. Of particular interest is a subset of networking modules, which facilitate the request and exchange of musical control data over a local-area network.

The networking functions of N.R.C.I. can be subdivided into two protocols, *request* and *command*, which are employed to meet different needs within the group. The *request protocol* enables unplanned exchange of control data to take place, as instigated by the receiver. Under this protocol, parameters are limited to four perceptually strong data types, these being pitch, amplitude, duration, and rhythmic onset. The *command protocol* offers an alternative method of interconnection, where parameter exchanges are driven by the sender, and may be used to take control of a specific target machine. In contrast to the limited data types of the request protocol, the *command protocol* may be used to exchange a parameter of any type, but the sender and receiver must agree on a descriptive name beforehand. As such, whilst it is possible that this kind of connection could be established during the course of a performance using extra-musical communication channels (e.g. a chat protocol), it can said that this protocol is inherently better suited to pre-planned situations

[13] https://ccrma.stanford.edu/~cburns/NRCI/

[14] http://www.puredata.info

where the performers are known and the terms of the improvisation have already been established, rather than truly impromptu situations.

Overall, N.R.C.I. presents a practical, workable solution for networked parameter sharing between co-located laptop performers. In terms of inclusivity, the toolkit approach is a valid one as it encourages users to develop their own approach, and the use of PureData addresses the technical demands of an inclusive approach through cross-platform operation[15]. However, as with the M.P.G. Carepackage, N.R.C.I. is primarily intended for use within an existing ensemble using a unified set of tools. As such, wider usage is restricted without further modification. One way of addressing this would be for non-PureData users to develop their own Open Sound Control[16] enabled solution to allow PureData to communicate with their application of choice. As seen previously, N.R.C.I. limits sharing to specific parameter types within the more easily accessible request protocol. However, this is more of a convention than a hard-and-fast standard, and as such is less problematic than the approach taken within the M.P.G. Carepackage. For example, users could choose to map the default pitch parameter to control a filter cut-off, thus overriding the designer's original intentions regarding the type of musical communication taking place.

In terms of the potential mutability of the underlying model of interconnection, N.R.C.I. proves to be more developed than the M.P.G. Carepackage, in that players may individually request or take over the parameters of other users. However, these may still only be used to create one-to-one or one-to-many mappings, with no higher-level control or visual feedback provided as to the status of group interconnections. With respect to the network infrastructure employed, communications within the N.R.C.I. system take place over a single UDP port, in a star network topology, with the advantage of a decentralised system being that *"if any one performer experiences a crash, the rest of the performers can still take full advantage of the network"* (Burns and Surges 2008). Finally, the use of a broadcast protocol, whilst easy to manage, can be criticised conceptually for inefficiency, as non-requesting receivers of the broadcast stream must discard unwanted data, wasting both communications bandwidth and processing power.

[15] PureData is available for the Windows, Linux and MacIntosh OS X platforms.

[16] http://www.opensoundcontrol.org

1.3.3 Bridge

Bridge[17] is a standalone software application developed by Mitani and Wyse at the Arts and Creativity Lab, Interactive and Digital Media Institute, National University of Singapore. The aim of the software is to manage connectivity between distributed applications and/or physical devices. To achieve this, users connect through an intermediate java-based application in order to share their inputs and outputs with one another. The end goal of the system is, in the words of the authors, to manage *"addressing and mapping between components making instrument design and networked compositions more object oriented, reconfigurable, and portable."* (Wyse and Mitani 2009).

Of the systems under consideration, Bridge is the most inclusive in terms of achieving unrestricted cross-platform and cross-application operation. The system is able to translate from and to a number of different protocols including OSC (via UDP), TCP sockets, and MIDI, making it highly flexible in terms of integration with almost any sound-producing software or hardware a performer could provide. In addition, some of the more social aspects of inclusivity are also addressed through a dynamic approach to networking, which allows users to join or leave the system at any time.

In terms of mutability of topology, any model of interconnection can be collected as a scene and shared. According to the authors, *"[t]his lends itself to interesting improvisational possibilities as well as compositional structuring. For example, the ensemble could navigate through a sequence of musical sections each embedded in a different network architecture"* (Wyse and Mitani 2009). If we consider use of the Bridge system based on Winker's models of performance (as discussed in Section 1.2.4), it is possible to envisage a situation where the system performs a role analogous to that of a score in a *conductor* or *chamber music* model. However, the system proves to be less workable if we imagine a situation based on the *improvisation* or *free improvisation* models, which are characterised by the spontaneous interactions. The key limitation of the Bridges system at the current time is the lack of a truly intuitive interface with which to manipulate the underlying network topology. Currently a patchbay-style view is provided, where connections can be created and removed dynamically on a connection-by-connection basis, but only after a reasonably elaborate set-up process has taken place. This makes it difficult for meaningful interactions to be established quickly enough to be useful in improvised performance situations. This could be

[17] http://bridge.anclab.org/

addressed by providing some higher-level conceptual representation of (and control over) the model itself.

1.3.4 Digital Orchestra Toolbox

The Digital Orchestra Toolbox[18] (D.O.T.) is a project headed by Joseph Malloch, Stephen Sinclair, and Marcelo M. Wanderley of McGill University in Montreal, Canada. The purpose of the toolbox is to support further research into interaction and performance within the wider Digital Orchestra project. As such, the toolbox shares many design goals with the previously discussed Bridges and N.R.C.I. projects.

The project successfully addresses issues of inclusivity by providing a plug-and-play network environment, to which controllers and synthesizers can announce their presence in order to make their input and output parameters available for arbitrary connection. These connections are managed using an OSC-controlled graphical mapping tool, implemented in Max/MSP, which allows gestural data streams to be dynamically connected and modified, making the overall interconnection topology highly mutable.

Like N.R.C.I., D.O.T. takes a modular approach to enabling network communications, with the toolkit itself coded as a set of Max/MSP abstractions. These contain many useful functions for handling network registration, as well as mapping and automatic scaling of data.

In terms of network infrastructure, the D.O.T. developers note the inefficiency of using a shared bus for all network communications, as implemented in the N.R.C.I. system. Instead, a common 'admin-only' bus for device announcement and resource allocation is used, with subsequent communications occurring directly between players via UDP.

In summary, the D.O.T. shares many benefits with the toolkit approach found in N.R.C.I., but with the added benefit of a more efficient network infrastructure and more open implementation. In addition, the D.O.T. graphical user interface proves easier to manipulate than the Bridges system, but questions still remain regarding its use as a creative tool in the course of performance.

[18] http://www.idmil.org/software/digital_orchestra_toolbox

1.3.5 Summary of Findings

The systems considered in the previous sections can be said to fall into two categories, with Bridges and D.O.T. proving to be the more open-ended, in contrast to the do-it-yourself approach of N.R.C.I. and the M.P.G. Carepackage, which impose greater restrictions on use. It can also be said that these latter systems tend to rely on establishing connections between players via chat protocols, physical cues or verbal communication, rather than by manipulation of a model of interconnection.

A significant advantage of both the Bridges and D.O.T. system is their ability to host diverse approaches to performance by supporting a wider range of communications protocols, as well as allowing for a dynamic approach to interconnection. This is achieved in both systems using a configurable mapping layer, which can be altered during the course of a performance. In terms of possible topologies, whilst both the D.O.T. and Bridges systems support movement between a range of models of interconnection within a single performance, this largely favours predetermined situations and neither system allows for weighted control of parameters as opposed to creating direct on/off connections. Finally, it is important to note that neither of these systems provides an intuitive graphical representation of interconnection or a high level method of manipulating the underlying network topology.

1.4 Aims of an Open-Ended Parameter Sharing System

To address the key areas of development highlighted by the previous section's findings, a number of aims are proposed for developing an open-ended parameter sharing system.

1.4.1 Inclusivity

In order to be termed inclusive, the system should retain players' existing performance practice as far as possible, in that it should:

a) enable players to quickly and seamlessly share a parameter with the group,
b) integrate with players' existing software and hardware, placing no restriction on the approach taken to sound creation,
c) allow players to join and leave a performance at any time,
d) demonstrate potential for flexible use in a range of open and closed performance situations (e.g. in pre-composed or improvised situations).

1.4.2 Mutability

As previously stated, the mutability of a system can be defined as the ability of players to manipulate the underlying model of interconnection during the course of a performance. To achieve this, players should be able to:

a) visually understand the current model of interconnection that applies to the group,
b) manipulate the model to generate a wide range of interconnection topologies.

2. System Design and Implementation

2.1 Towards a Mutable Model of Interconnection

This section charts the development of a mutable model of interconnection, moving from first principles through to a fully manipulable graphical implementation in software. Initial work focuses on establishing a number of fundamental rules which govern how a common parameter type can be placed under shared control (Section 2.1.1). This leads to the identification of a number of archetypal models of interconnection (Section 2.1.2) and the development of an appropriate graphical representation for representing these archetypes (Section 2.1.3). With this complete, focus then shifts to the mutability of the model, which relies on establishing a number of operational transformations (Sections 2.1.4 and 2.1.5) and implementing these using the F.T.M. extensions to Max/MSP (Section 2.1.6). To complete the implementation, it is necessary to place these manipulations under the direct control of players (Section 2.1.7).

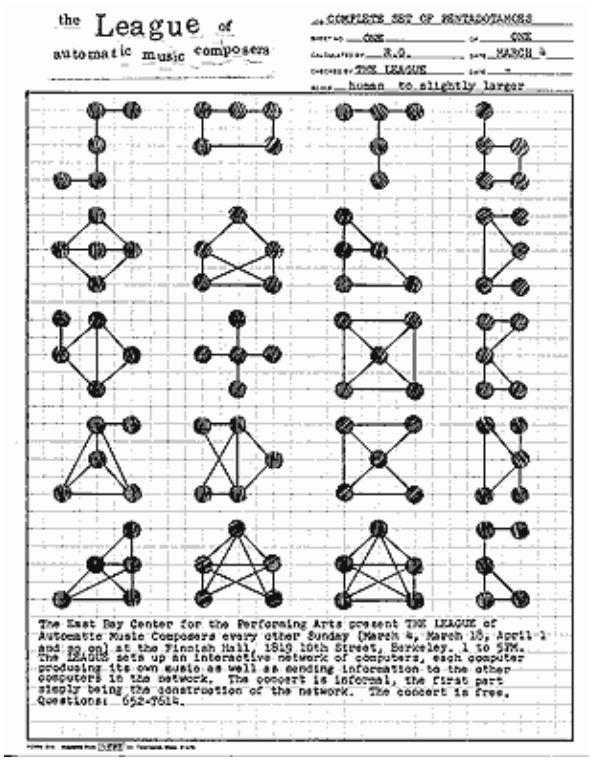


Figure 3: Flyer from a Spring 1979 concert by the League of Automatic Music Composers

The initial inspiration for developing a reconfigurable network for musical communication was provided by the League of Automatic Music Composers, whose early promotional

material shows a range of possible models of interconnection between five group members (Figure 3). In a more recent study, Weinberg (2005) proposes an *asymmetric weighted flower topology* (reproduced in Figure 4), whereby *"a weight system can [...] be assigned and controlled in real time to provide dynamic levels of influence*". By placing such a weight system under group control, it is possible to move between a wide range of models of interconnections during performance, including those in which players share control of each other's parameters.

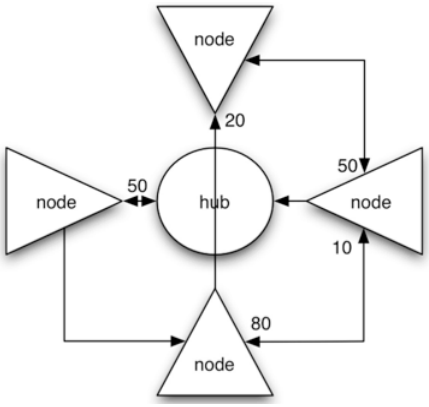


Figure 4: Weinberg's *"asymmetric weighted flower topology",* consisting of a series of weighted gates (2005)

2.1.1 'Without a Trace': The Fundamentals of Sharing Control

In order to understand how synchronous parametric control can be shared within a group, it first makes sense to consider a series of players who each control a parameter of their own. Presuming that each of these parameters is of the same type, this 'disconnected' situation can be seen as a starting point from which players can pool their influence. If we consider this pool as a matrix consisting of rows in a table, we can begin to map out ways in which the total group influence might be shared between its members. A basic rule here is that the total amount of influence available must always sum to the number of players participating, as shown at the bottom right hand corner of the weights tables in Figure 5.

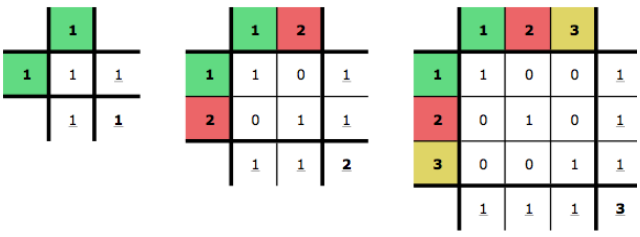


Figure 5: Weights tables showing the joining process for a succession of three players

On an individual level, the fact that influence is pooled means that each player's parameter must always be under the control of some player, or combination of players, within the group. In terms of the weights table in Figure 5, this is shown by the fact that each column adds to one. This highlights the push-pull nature of control, where if influence is taken away from one player, it must be given to another. This can be regarded as a fundamental rule which must be followed in order to avoid 'free-floating' situations where no-one is in control.

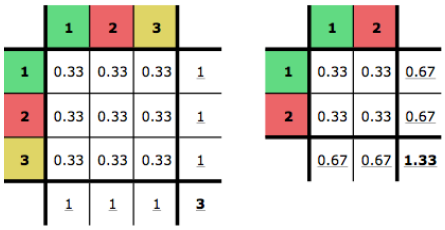


Figure 6: Weights tables highlighting a problem in the leaving process

Conceptually, the process of joining the sharing process presents no problem, as players begin by controlling only their own parameter. Figure 5 illustrates this, showing three players joining one by one with all columns adding to one at all times. However, the process becomes more difficult to manage when a player wishes to stop sharing, as to avoid an imbalance, they must then remove any influence they have over other players' parameters and return to a 'disconnected' situation. Figure 6 shows the imbalance that occurs when a player leaves whilst still holding influence over others in the group, where the individual columns no longer add up to one. These two processes of joining and leaving 'without a trace' can be considered as fundamental rules which should be observed when sharing parametric control.

2.1.2 Identifying Archetypal Models of Interconnection

Paying attention to the rules identified in the previous section, it is possible to define three scenarios, which can be seen as archetypal models of interconnection and which can then be combined to define more complex cases.

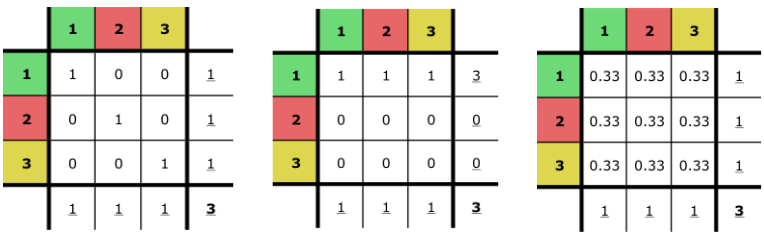


Figure 7: Archetypal independent (7a), dominant (7b) and shared (7c) models

In the scenario shown in Figure 7a, each player is restricted to controlling only themselves. As we have seen previously, such a scenario is analogous to a disconnected situation and defines a starting point from which control may be shared. We can term this an *independent* model.

In the next scenario in Figure 7b, Player 1 holds total control over all three players' parameters, while the other players remain subordinate, unable to influence either themselves or each other. If we imagine the shared parameter as pitch, and Player 1 as sending a middle C note value, all players will be provided with the same middle C value at their pitch output. We can refer to this as a *dominant* model.

In the third scenario, shown in Figure 7c, each player has only a partial level of influence over their own parameter, but also has the same amount of influence again over the parameters of the other players in the group. This represents a shared-mind situation, where players hold a fractional degree of control, in conjunction with one or more of their peers. We can term this a *shared* model.

To give a practical example of a *shared* model, if each player's input is a middle C note value, the resulting output on all machines will also be a middle C value. However, if players' inputs are all different (which is likely to be the case), then the result will be a note value which lies somewhere between the inputs provided but belongs to no single player. In terms of responsiveness, the shared model gives each player a fractional amount of control over each player's overall pitch range, with the exact amount of influence depending on the number of players in the network.

It is worth pausing at this point to consider the previously identified archetypes in terms of their potential perceptual strength, as Burns and Surges have considered within their work (2008). In this respect, it can already be suggested that the individual and dominant models are likely to prove perceptually strong, whereas the results of the shared model will likely prove more difficult to recognise, as players are restricted in terms of their overall control and influence, which is not localised in a single body. These features may be exacerbated as the number of peers sharing control increase and the amount of influence held by each player gets diluted. It will be interesting to note if this leads to shared models being generally less desirable to set up or whether they might take on another role which has not yet been anticipated.

2.1.3 Representing Interdependencies Visually

With an initial set of archetypes defined, it became important to consider how these might be represented visually in an intuitive manner for players. To this end, a conceptual study was undertaken, involving development and comparison of a number of initial representations. The results of this process were then used to implement a flexible graphical model in Max/MSP. Here we present a number of examples taken from this model. These are closely linked to the final representation employed in the graphical user interface, but at this stage the model only takes account of how interconnections are displayed, not how they might be applied or manipulated in the course of performance. Examples of the integration of this flexible model into the G.U.I. itself can be found in Section 2.3.

In the initial design shown, peers are represented as circular nodes arranged at equistance from a central point. Each peer is assigned a unique identifying colour, which permanently defines the inner core of their node and represents a single body of influence. In contrast, the outer section of the node shows the balance of influence currently held over this body, which may be made up of one or more colours to indicate that it is being shared by one or more players. Figure 8 presents this representation for each of the previously identified archetypal models.

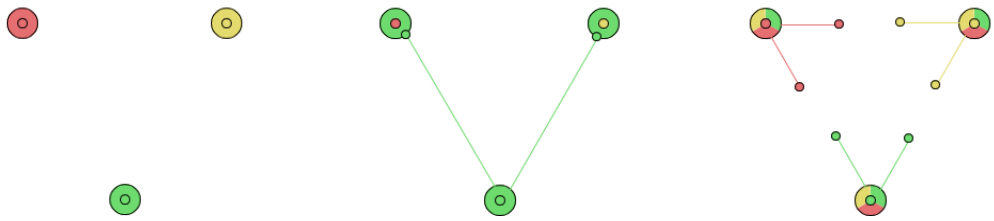


Figure 8: Conceptual representation of independent (8a), dominant (8b) and shared (8c) archetypes

In an independent situation (Figure 8a), the inner and outer sections of each node are shown in the same colour, indicating that each peer holds sole control over their parameter. In a dominant situation, however, the colour of the controlling player (in this case green) fully occupies the outer section of each node (Figure 8b). In contrast, shared situations can easily be recognised by the segmentation of the outer section into more than one colour (Figure 8c).

Whilst this pie-chart style representation provides all the necessary information for users to understand the exact model of interconnection being applied, and is able to represent proportional levels of influence with reasonable accuracy, it seemed unintuitive in terms of

providing an overview of who each player is controlling at a given time. To address this, it was decided to add interconnecting lines between players in order to represent the outgoing level of influence of each player. The use of two systems for representing levels of influence was initially a concern, but over time these have been found to reinforce each other, emphasising local and remote influence respectively.

Whilst representing connections using lines seems like an obvious choice, a number of different methods were considered for achieving this. Initially, for example, each line always spanned the full range between the controlling and the controlled player and the amount of influence was shown by the transparency of the line. This was soon rejected however, as it proved difficult to accurately compare the relative luminosities of colours and as connections overlapped, colours combined to produce new meaningless shades. In the end, line length proved to be a far better fit in terms of being able to quickly identify the amount of influence being applied. In the representation shown, the lines themselves have been kept in the controlling player's colour but are accompanied by terminating circles, filled with the same colour, which help to emphasise the precise amount of influence held and to allow changes to be tracked more easily. Within this basic design, care has been taken to ensure that reciprocal connecting lines between players do not overlap, allowing for clear identification of each connection (Figure 9).

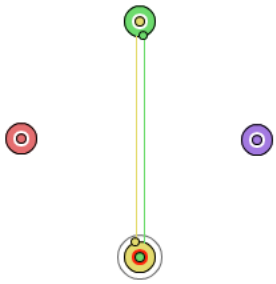


Figure 9: The chosen representation allows for reciprocal non-overlapping connections

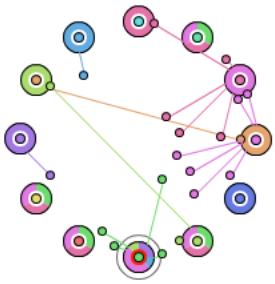


Figure 10: Conceptual representation of a complex twelve-player network

Finally, a key consideration here was the suitability of the design for representing larger groups. In practise, the limit was placed at twelve players (Figure 10), which is seen as more than adequate within the scope of the project. With many more players than this, the model begins to require more screen space, and subtle differences between colour shades makes the identification of players more difficult.

2.1.4 Creating Complex Models of Interconnection by Manipulation

With a basic conceptual representation complete, it was possible to return to the archetypal models identified in Section 2.1.2 in order to try to draw out their essential characteristics and consider how each might be combined to create more complex networks of interconnection. In turn, these networks can be used to evaluate whether an intuitive conceptual representation is maintained in complex situations.

When considering each archetypal model from the perspective of a single player, we can see that for each the key questions are a) 'How much control does a player have?' and b) 'Over whom?' These can be defined as the *amount of influence* and *location of influence* respectively. In regards to the former, a player's level of influence may range from non-existent (i.e. the player is subordinate) to total control over the whole group (i.e. the player is dominant), whilst the latter may be categorised in terms of being either local or remote.

2.1.4.1 Independent Model

If we take the independent model as a starting point (Figure 11) and transpose each player's location of influence from local to remote, we create a phantom situation where each player no longer controls themselves, but instead controls another member of the group (Figure 12). This kind of movement from local to remote is one way in which the model of influence may be manipulated, which may be defined as an *exchange* operation. As opposed to independent, this kind of network can be defined as *individual,* where influence is still held over a single body in total, but one which is no longer local to the controlling player. Figure 13 shows a larger *individual* network, which can be defined in terms of a local and remote mix of influence occurring in single bodies.

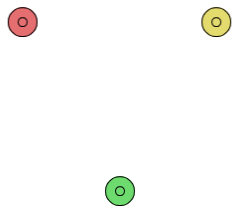


Figure 11: The independent archetype as a starting point for transposition of influence

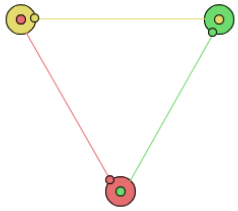


Figure 12: The exchange model, representing transposed location of influence

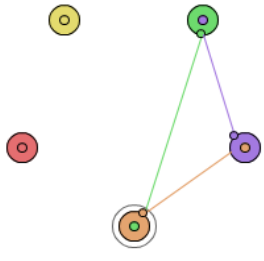


Figure 13: Complex five-player mix of independent (local) and individual (remote) archetypes

2.1.4.2 Dominant Model

In contrast to independent/individual models, the dominant model is characterised by a disparity in the amount of influence held by each player, which gives one player full control and makes the others subordinate (Figure 14). An exchange of influence from this position may involve either the total influence of the player (Figure 15), which leads to this player assuming dominance, or to transposition of only the influence relevant to the local player (Figure 16). This leads to a hybrid *independent-dominant* model, where Player 1 continues to exercise control over their own parameter and that of Player 3, whilst Player 2 takes back control of their own parameter. This hybrid of the dominant and independent models can be replicated in a more complex form (as shown in Figure 17). Again, both kinds of substitution prove to be useful transformations in terms of developing a mutable model of interconnection.

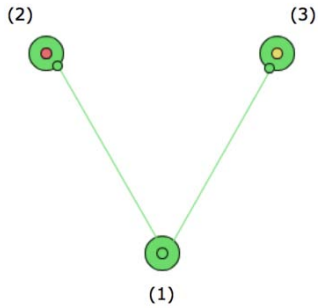


Figure 14: The dominant archetype (Player 1 has all influence)

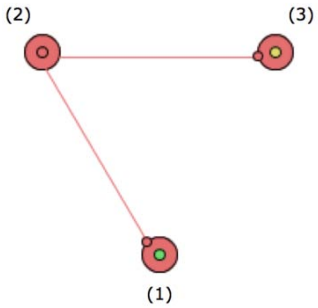


Figure 15: Alternative dominant archetype (Player 2 takes all influence)

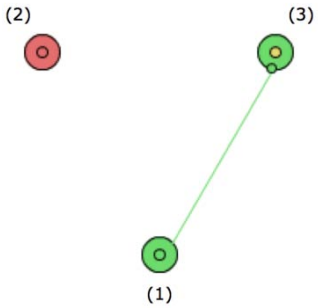


Figure 16: Complex independent-dominant model (Player 2 takes local influence)

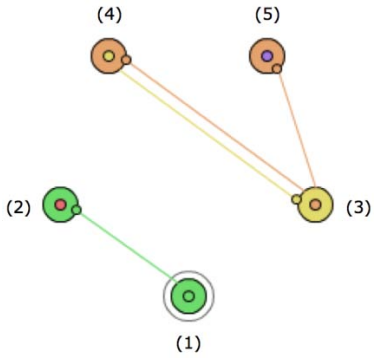


Figure 17: Complex five-player mix of independent and dominant archetyes

2.1.4.3 Shared Model

As with the independent model, a shared situation is typified by equality in the total amount of influence held by each player. The difference here is that, rather than this influence being located in a single body, it is instead distributed equally amongst the players in the group. Therefore we can further describe this model as being *equally* shared.

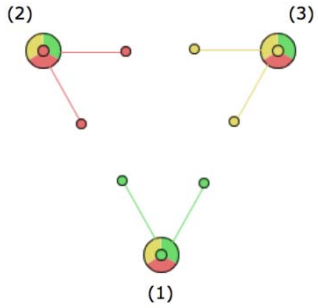


Figure 18: Shared archetype, where each player has equal amounts and locations of influence

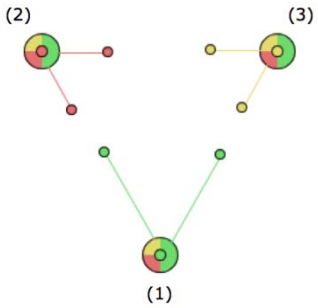


Figure 19: Unequal shared model, suggesting movement towards dominance by Player 1

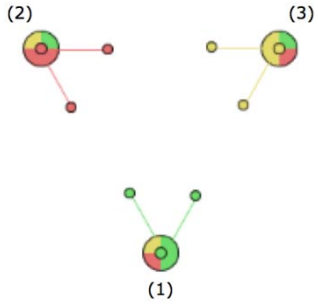


Figure 20: Unequal shared model, suggesting movement towards independence by all players

Taking an *equally shared* model as a starting point (as shown in Figure 18), it is easy to see that any transposition of influence will have no effect, as both the location and amount of influence are equal for all players. However, there are other possible directions of movement that may prove more valuable. Firstly, it is possible for a player to move towards a dominant position (Figure 19, Player 1), which in turn pushes other players towards a subordinate position (Figure 19, Player 2 or 3). Secondly, players may move towards independence (Figure 20), which is marked by the redistribution of disparate influence within a single body.

2.1.5 Making Models Mutable

In the previous section, we have seen how complex models of interconnection can be created by manipulating simple archetypes. Here we define four specific operations, in terms of the type of influence transfer taking place in each case. These are termed *take*, *give*, *exchange* and *share* and are further described in Table 2. For each operation, it is required to define a single source player (i.e. the local player who will initiate the transfer), one or more target players (i.e. the remote player or players) and the location of the influence involved in the transfer. This latter aspect is perhaps the most difficult to grasp, with Table 3 providing an overview of the three options that have been made available to

players. The most important distinction to note is between the *local* and *all* options. For example, taking *all* influence from all players would lead to a dominant position, whereas taking *local* influence from all players will always lead to the local player taking back complete control of their own parameter.

Operation	Description	No. of Target Players
Take	Take influence from remote players	One or more
Give	Give local influence to remote players	One or more (selection of multiple targets will share influence equally between them)
Exchange	Exchange influence with a remote player	One
Share	Share local influence with remote players	One or more (as for *Give*, but a share of the influence is retained for the source player)

Table 2: Overview of operational manipulations, describing the transfer of influence that takes place in each case

Location	Description
Local	Only influence relevant to the local player will be involved in the operation
All Remote	Only influence not involving the local player will be involved in the operation
All	Influence over both local and remote players will be involved in the operation

Table 3: Overview of the locations of influence which can be involved in transfer operations

One of the features of the operational approach outlined above is that it is sequential in nature, meaning that transformations often consist of multiple steps. At this stage the fact that operations are applied in stages does not present a problem, as the purpose of the target model is to set up an idealised model of interconnection rather than to apply transformations directly in real-time. In fact, this is an oft used approach in sequential systems (Weinberg 2005) which allow transformations to evolve from an existing starting point.

2.1.6 Implementation Using F.T.M.

An initial mutable model was implemented using the F.T.M. extensions to Max/MSP, as developed by the Real-Time Musical Interactions team at I.R.C.A.M[19]. These objects extend

[19] http://ftm.ircam.fr/

the basic data structures available in Max in order to support complex, multi-dimensional types (Schnell et al. 2005). The *fmat* structure in particular is well suited to representing models of interconnection as two dimensional floating point weight matrices (such as those shown previously in Figure 7). Beyond the structures themselves, the real benefit of using F.T.M. here is that it provides a general set of transformation functions for each data type that are readily applicable to the task of manipulating a model of interconnection[20]. These functions can be applied to whole matrices, but also to specific elements within them, such as row or column vectors. Therefore, by applying simple functions such as *fill* in both these ways, it is possible to quickly develop algorithms that implement the types of transformations described in the previous section. In addition, the ability to iterate over the rows and columns of matrices facilitates the implementation of operational transformations such as substitution or transposition of influence between players.

2.1.7 Exposing Control

Having established a mutable model of interconnection, it now became necessary to explicitly define how control of the model would be exposed to members of the group, in order to afford a fluid and manageable experience in a performance context. Conceptually, it was of prime importance to separate off the creation of new models from the process of applying them in real time. This was necessary so as to allow each player to arrive at their own ideal model via multiple operational manipulations, without affecting (at each step) the model currently being used in performance. In order to clarify this split for performers, it was chosen to represent these two types of model as entirely separate elements of the graphical user interface, which are referred to as the *live* and *target* models, respectively. In order to apply a target model, a player must first take control of the live model. In the chosen implementation, only one member of the group may be in control at any one time, which is clearly communicated by a) 'greying out' the appropriate section of the user interface for all but the controlling player and b) prominently displaying their name and colour on the user interface. This stipulation was made in order to prevent multiple players fighting for control by trying to each grab the same controls at the same time.

Once a player is in control, the target model can be applied to the live model in a number of ways. The simplest way is switch instantaneously to the new model from the old one. This

[20] While the Jitter component of the Max environment provides support for similar matrix structures, these are primarily oriented towards video processing tasks.

may be desirable in some situations, but there are also options to automatically ramp or manually scrub between the two models to create evolving transitions. This is achieved by a process of linear interpolation between the live and target matrices. Further user-centred discussion of all these features can be found in Section 2.3.

2.1.8 Summary

Within this section, I have established a number of fundamental rules which enable synchronous sharing of a common parameter type to take place. These have been presented as part of a wider framework of interconnection which encompasses both graphical representation and operational manipulation. The chosen implementation represents one approach to collaborative control, which facilitates the independent creation of interconnectional models by each player, whilst at the same time imposing a degree of constraint on how these are applied during live performance. The resulting system enables a wide range of possible models of interconnection to be explored during the course of a performance.

2.2 'Is There Anybody Out There?': Establishing a Network Infrastructure

This section presents a number of fundamental choices that needed to be made regarding the underlying network infrastructure used in the project. The end goal of this work was to implement a *peer transport*, in order to facilitate targeted delivery of messages to any combination of players registered on the network. This feature underpins many important distributed functions of the patch, ranging from the transmission of data regarding the current state of the live model (sending the same information to all peers), through to stateful retrieval of each player's current level of influence when a new peer registers on the network (requesting different information from each peer).

To reach this point, a system is first required that allows users to register with each other on an ad-hoc basis and which is able to maintain an accurate picture of the number of players present on the network at any one time, on each users machine. The core aims here are a) reliability, b) automatic configuration (users should not have to provide additional information about their networking hardware in order to register) and c) inclusivity (the ability for players to join and leave at any time).

In the projects surveyed in Section 1.3, a number of different solutions were found to problem of registering users on the network. This variation can be attributed to a shortage of standardised software extensions (i.e. externals for Max/MSP and PureData), which address the larger problem of establishing communications channels between peers. Whilst some pre-packaged solutions do exist[21], ultimately it was decided to reject these and instead to pursue a native implementation using existing Max/MSP networking objects[22] (Neville 2006). This approach guarantees re-usability in future projects and is in line with similar choices made in the Max/MSP-based projects already considered, such as the M.P.G. Carepackage (Wolek 2010) and Digital Orchestra Toolkit (Pestova et al. 2009).

2.2.1 Advantages of a Decentralised Approach

Despite the increased complexities, an early decision was made to adopt a decentralised, peer-to-peer approach in terms of the overall network infrastructure. Whilst a centralised client-server model has some benefits in terms of the need to only maintain an accurate picture of the network on one machine, there are a number of key disadvantages to this method. Firstly, centralisation provides an additional point of failure, which proves less than ideal in a performance situation. Secondly, and perhaps equally importantly, centralisation places an extra burden on one member of the group in order to maintain and run the system. In a climate where users often have their own complex setups to manage, which may consist of using multiple applications within a single performance, this extra level of responsibility is seen as a significant burden. As such, a decentralised approach, which employs an identical registration system for all players, can be seen as more compatible with the principles of inclusive networking.

2.2.2 Overview of the Registration Process

The basic registration process can be summarised in three stages, with reference to the specific Max/MSP abstraction developed. Figure 21 provides a timeline overview of this process, illustrating the case of a new player (named Scott) registering on a existing network which contains two players (named Adam and Sam). The figure also includes labels which indicate the processes that circumscribe each abstraction.

[21] The OSCbonjour externals for Max/MSP (Muller, 2006) were investigated for managing zero configuration registration, but were found to be unreliable in practice.

[22] The only exception being the use of Open Sound Control route object (OSC-route).

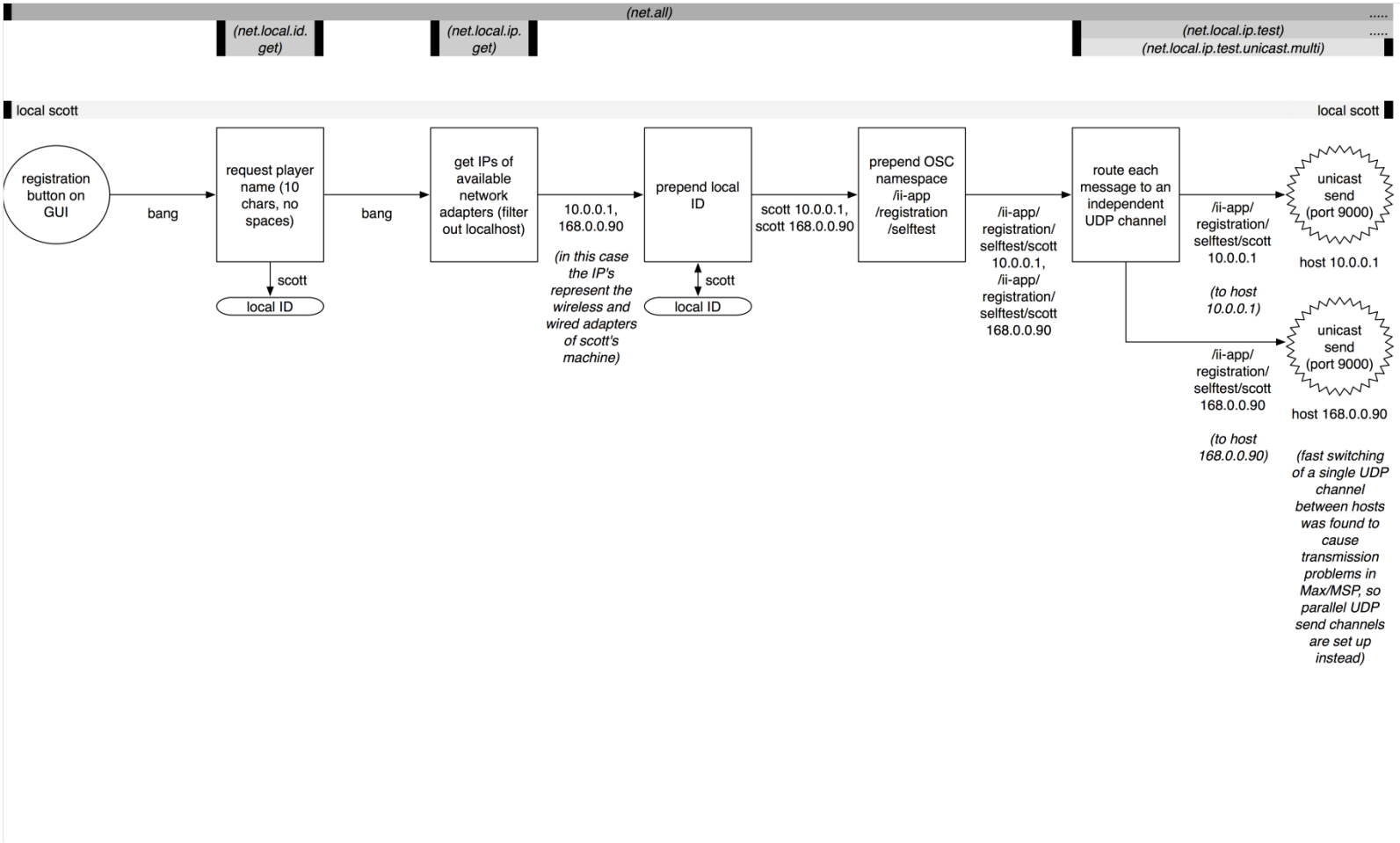
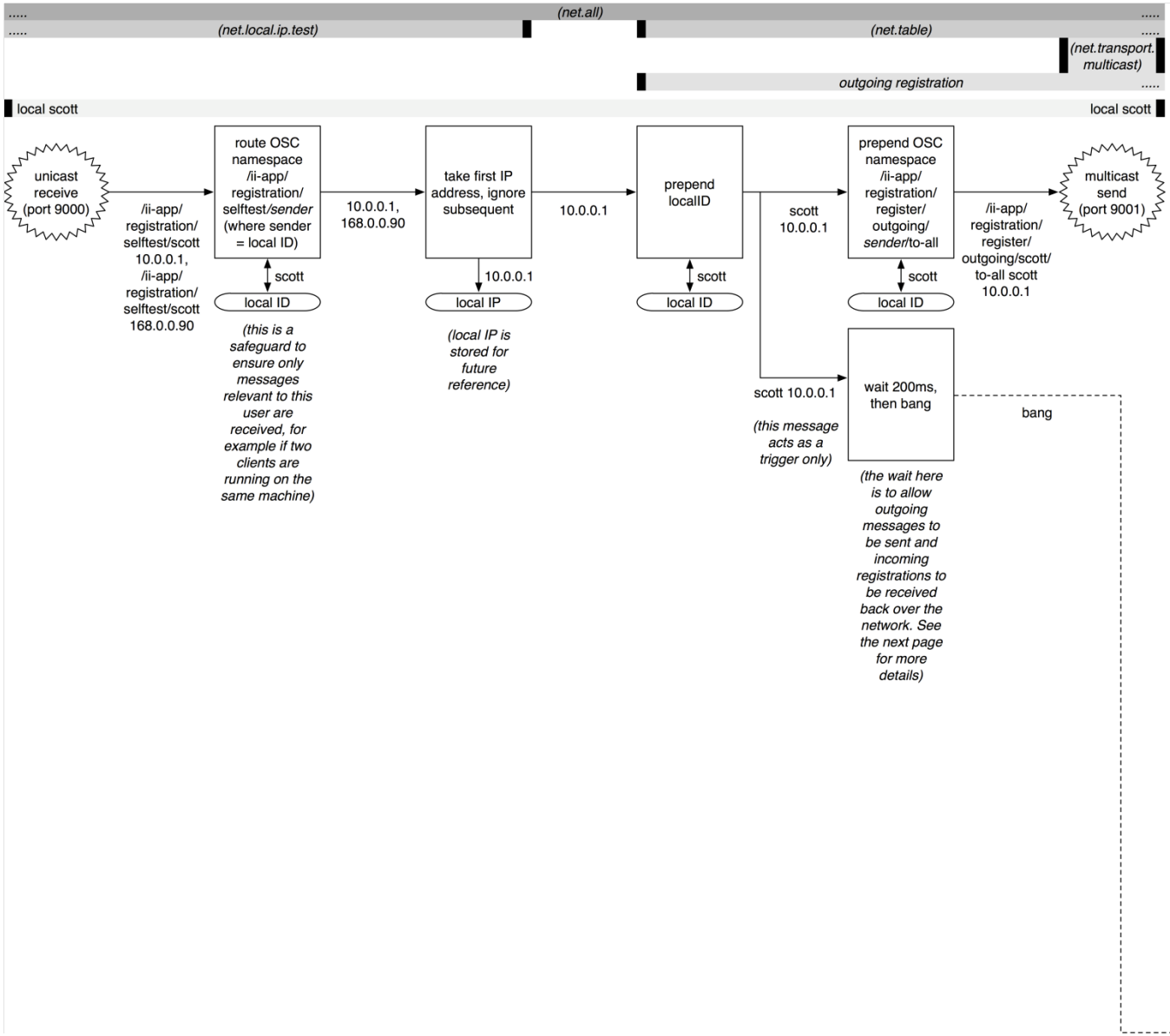
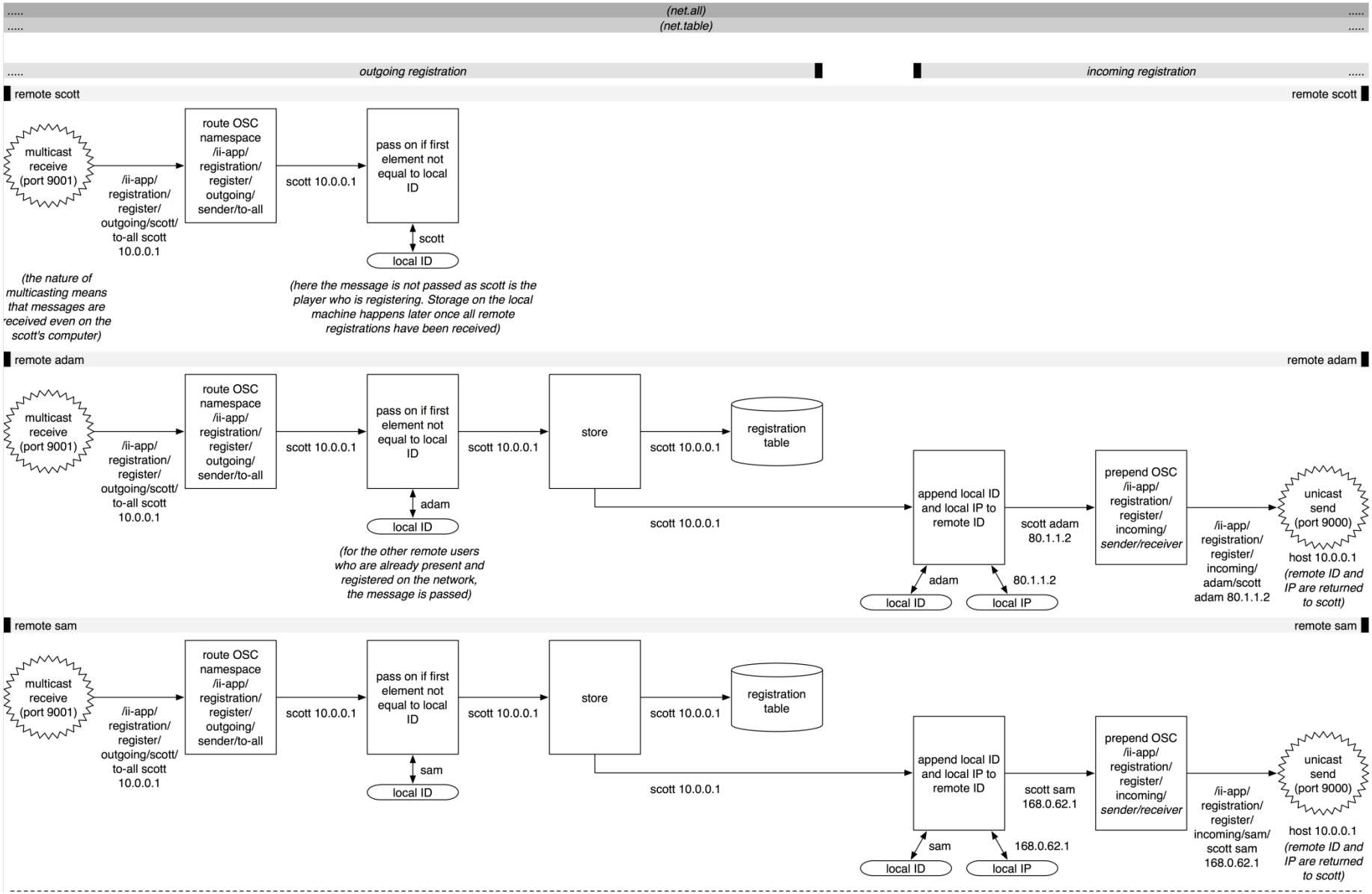
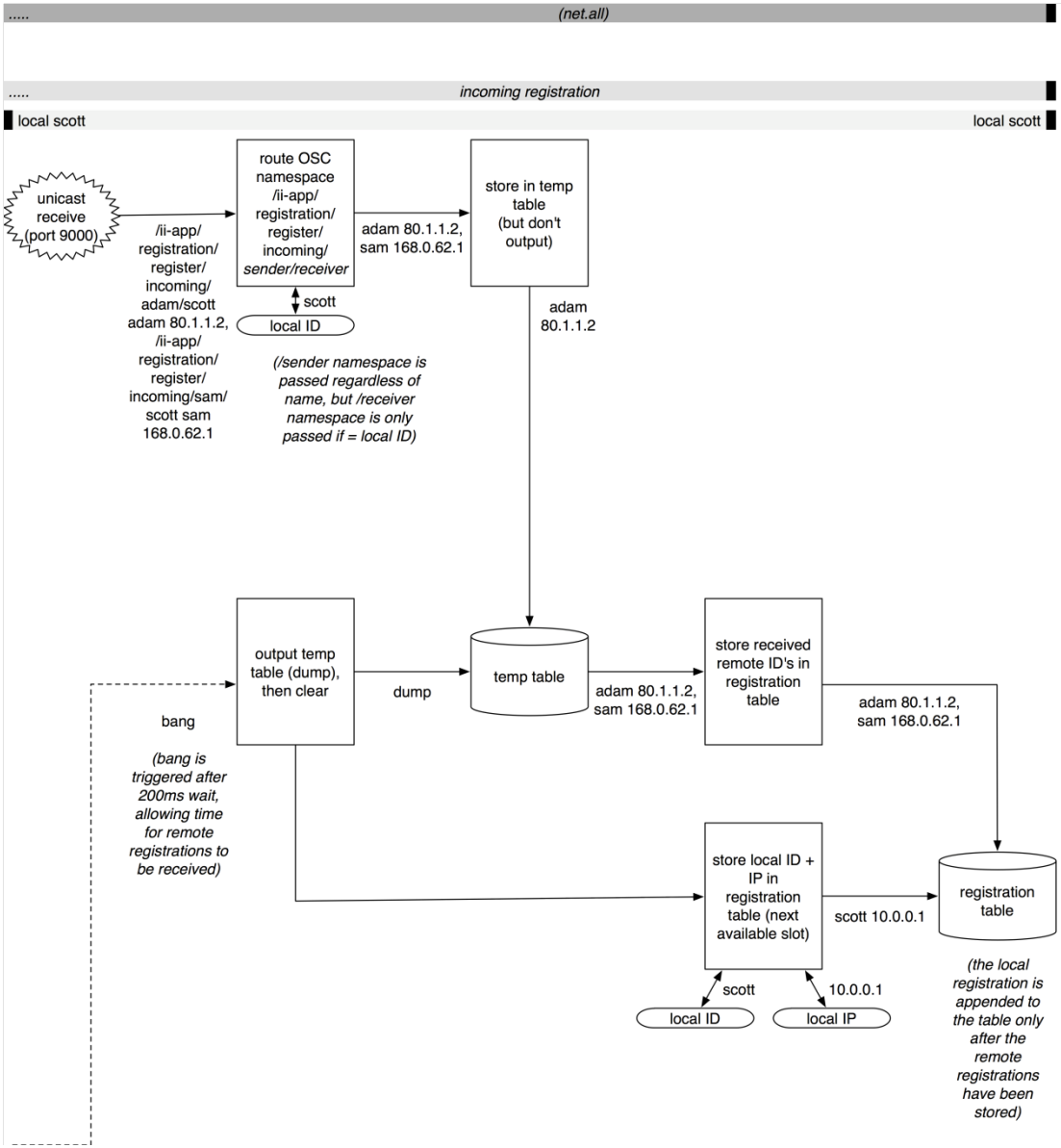


Figure 21: Timeline diagram of the network registration process (continues on subsequent pages)


(net.all)
(net.local.ip.test)
(net.table)
(net.transport.multicast)
outgoing registration
local scott
local scott
unicast receive (port 9000)
/ii-app/ registration/ selftest/scott 10.0.0.1, /ii-app/ registration/ selftest/scott 168.0.0.90
route OSC namespace /ii-app/ registration/ selftest/sender (where sender = local ID)
scott
local ID
(this is a safeguard to ensure only messages relevant to this user are received, for example if two clients are running on the same machine)
10.0.0.1, 168.0.0.90
take first IP address, ignore subsequent
10.0.0.1
local IP
(local IP is stored for future reference)
10.0.0.1
prepend localID
scott
local ID
scott 10.0.0.1
(this message acts as a trigger only)
prepend OSC namespace /ii-app/ registration/ register/ outgoing/ sender/to-all
scott
local ID
/ii-app/ registration/ register/ outgoing/scott/ to-all scott 10.0.0.1
multicast send (port 9001)
scott 10.0.0.1
wait 200ms, then bang
(the wait here is to allow outgoing messages to be sent and incoming registrations to be received back over the network. See the next page for more details)
bang


(net.all)
(net.table)
outgoing registration
incoming registration
remote scott
multicast receive (port 9001)
/ii-app/ registration/ register/ outgoing/scott/ to-all scott 10.0.0.1
route OSC namespace /ii-app/ registration/ register/ outgoing/ sender/to-all
scott 10.0.0.1
pass on if first element not equal to local ID
scott
local ID
(the nature of multicasting means that messages are received even on the scott's computer)
(here the message is not passed as scott is the player who is registering. Storage on the local machine happens later once all remote registrations have been received)
remote adam
multicast receive (port 9001)
/ii-app/ registration/ register/ outgoing/scott/ to-all scott 10.0.0.1
route OSC namespace /ii-app/ registration/ register/ outgoing/ sender/to-all
scott 10.0.0.1
pass on if first element not equal to local ID
scott 10.0.0.1
store
scott 10.0.0.1
registration table
adam
local ID
(for the other remote users who are already present and registered on the network, the message is passed)
scott 10.0.0.1
append local ID and local IP to remote ID
scott adam 80.1.1.2
prepend OSC /ii-app/ registration/ register/ incoming/ sender/receiver
/ii-app/ registration/ register/ incoming/ adam/scott adam 80.1.1.2
unicast send (port 9000)
host 10.0.0.1
(remote ID and IP are returned to scott)
adam
80.1.1.2
local ID
local IP
remote sam
multicast receive (port 9001)
/ii-app/ registration/ register/ outgoing/scott/ to-all scott 10.0.0.1
route OSC namespace /ii-app/ registration/ register/ outgoing/ sender/to-all
scott 10.0.0.1
pass on if first element not equal to local ID
scott 10.0.0.1
store
scott 10.0.0.1
registration table
sam
local ID
scott 10.0.0.1
append local ID and local IP to remote ID
scott sam 168.0.62.1
prepend OSC /ii-app/ registration/ register/ incoming/ sender/receiver
/ii-app/ registration/ register/ incoming/sam/ scott sam 168.0.62.1
unicast send (port 9000)
host 10.0.0.1
(remote ID and IP are returned to scott)
sam
168.0.62.1
local ID
local IP


(net.all)
incoming registration
local scott
local scott
unicast receive (port 9000)
/ii-app/ registration/ register/ incoming/ adam/scott adam 80.1.1.2, /ii-app/ registration/ register/ incoming/sam/ scott sam 168.0.62.1
route OSC namespace /ii-app/ registration/ register/ incoming/ sender/receiver
scott
local ID
(/sender namespace is passed regardless of name, but /receiver namespace is only passed if = local ID)
adam 80.1.1.2, sam 168.0.62.1
store in temp table (but don't output)
adam 80.1.1.2
output temp table (dump), then clear
bang
(bang is triggered after 200ms wait, allowing time for remote registrations to be received)
dump
temp table
adam 80.1.1.2, sam 168.0.62.1
store received remote ID's in registration table
adam 80.1.1.2, sam 168.0.62.1
store local ID + IP in registration table (next available slot)
scott
local ID
10.0.0.1
local IP
scott 10.0.0.1
registration table
(the local registration is appended to the table only after the remote registrations have been stored)

2.2.2.1 User Input

(as handled by the *net.local.id.get* abstraction)

To activate the registration process, the user is prompted to enter a username of their choice. This is then used as an identifier for future communications. Whilst this is not guaranteed to be unique (other users may try to register with the same name), the benefit of using a user-entered ID over a randomly generated string is that it remains human-readable throughout. Crucially, this is the only input required by the user to register on the network.

2.2.2.2 Establishing an Incoming Connection

(as handled by the *net.local.ip.get* and *net.local.ip.test* abstractions)

This stage can be split into two separate processes. The first of these handles detection of available local network adapters, while the second tests each available adapter to determine if it can accept incoming connections. The core functionality for the detection of network adapters is already provided within Max/MSP by the *mxj net.local* object, with the *net.local.ip.get* abstraction simply re-purposing this output as a list. In addition, the localhost address (127.0.0.1) is filtered out of the results and an error message given if no adapters are detected.

The testing process (as handled by the *net.local.ip.test* abstraction) involves the set up of parallel UDP communication channels for each network adapter IP address. Test messages are then sent to each of these channels in quick succession, with the first message to be received determining the adapter that will be used. This use of parallel channels avoids the transmission problems associated with fast switching between host IPs when using the *udpsend* object in Max/MSP. If no message is received in a set amount of time, the test is deemed unsuccessful, and the user may check their setup and attempt to register again. It can be said that the main purpose of this testing process is diagnostic in nature, rather than being a true test of the capabilities of the network hardware itself. In other words, the reliability of the network connection is already presumed, with registration failures at this stage simply serving to prompt users to troubleshoot firewall issues or blocked networked ports. In practice, the process was found to be reliable and transparent, avoiding the need for users to select their adapter of choice on start-up. However, for best results, users should ensure they are connected to the network using only the adapter of their preference.

2.2.2.3 Registration

(as handled by the *net.table* abstraction)

Once an incoming IP address has successfully been identified, the actual process of registering the local user on the network can take place. This process can be considered in two distinct stages, which are labelled in Figure 21. The first stage, *outgoing registration*, requires the local peer to send information about themselves to be stored on each remote peer's machine. This is followed by a complementary process of *incoming registration*, where information about each remote peer is returned to the local peer and stored. Only after this process has taken place is the local peer registered on their own machine. By taking an *outgoing first* approach, the process requires only two transmissions per peer to build up an accurate picture of the network.

As can also be seen in Figure 21, differing network protocols are used for the outgoing and incoming stages of the process. This approach was informed by similar choices made in the M.P.G. Carepackage (Wolek 2010) and Digital Orchestra Toolkit (Pestova et al. 2009) projects. For initial outgoing registrations a multicast transport is used, which allows users to send their registration details to a group address, to be received by all users present on the same local network. This is a common zero configuration strategy, which elimines the need to know each player's incoming IP address prior to sending data. In contrast, at the incoming registration stage the target of the transmission is already known, so a specific UDP channel can be set up via which to return the remote player's details back to the local machine. Overall, this pairing of multicast and unicast communication protocols results in a more efficient infrastructure when compared to a multicast-only solution, with the trade-off being a more complex implementation.

2.2.2.4 Further Registration Complications

In practice, the simplified registration process outlined above is complicated by a number of additional factors, some of which are necessary to ensure robustness, whilst others add a degree of flexibility to the basic framework or exist to keep track of the order in which players register on the network.

One example of added flexibility is the use of sender and receiver namespaces. These are added to the end of existing OSC registration messages, which are then transmitted over the network and passed on locally if the receiver namespace is deemed to match the ID of the

local machine. For example, a typical incoming registration message taking place between the peers Dave and Bob would take the form:

```
/registration/incoming/dave/bob dave 192.168.4.1
```

In unicast situations where clients are located on separate computers, this process is redundant and exists only to allow human-readable tracking of messages across the network. However, the approach proves invaluable in allowing distributed operation of the software to be tested on a single machine, as it provides an extra layer of differentiation in situations where multiple peers are located on a single PC, thus ensuring that messages reach their correct destination.[23]

A final essential addition at this stage was the ability to keep track of the order in which users register, in order to assign unique index values to players, which are maintained across the group. Here, two types of index value are required, one which stays the same throughout the course of registration (referred to here as the *absolute index*), and another which changes to reflect the user's position relative to others (referred to as the *relative index*). These have significantly different uses, with the absolute index being used to set up communications channels and determine the colour of players (both of which need to stay the same throughout), whilst the relative index is used in order to determine where to insert new rows and delete old ones from weights matrices.

2.2.2.5 From Registration Table to Peer Transport

(as handled by the *net.peer.transport* abstraction)

Establishing an accurate picture of the players on the network is only one part of the process. Once registration has been successfully achieved, the information gathered must then be used to enable direct communication to take place between peers. This is the overall function of the *net.peer.transport* abstraction.

The abstraction itself operates by opening up parallel outgoing UDP channels based on each peer's username or numeric ID (as gathered at the registration stage). With this infrastructure in place, it is possible to send messages to individual users (by prepending

[23] See section i. of Appendix B for a video demonstration, as well as the *inclusive.interconnections.multi* patch in section ii. of Appendix A, which shows the final implementation.

them with either of these IDs), or the entire group (by prepending a '-1' message[24]). For example, in a three-player-network, where Tony is the local player, a message of 'dave pitch/value 0.1' sent to *net.peer.transport* would result in a the following output:

```
/peer-transport/pitch/value/tony/dave 0.1
```

In contrast, a message of '-1 pitch/value 0.1' would result in the following messages on each remote machine:

```
/peer-transport/pitch/value/tony/tony 0.1
/peer-transport/pitch/value/tony/dave 0.1
/peer-transport/pitch/value/tony/ellen 0.1
```

Additionally, the peer transport allows a global namespace to be specified, such as the name of the overall application within which the peer transport is functioning. For example:

```
/ii-app/peer-transport/pitch/value/tony/tony 0.1
```

Finally, it can be said that the implementation of the peer transport is essentially transport independent and could easily be adapted for use with TCP, if completely reliable (but less timely) communications are required.

[24] This method of operation was inspired by the MSP poly~ object.

2.3 Presenting the System

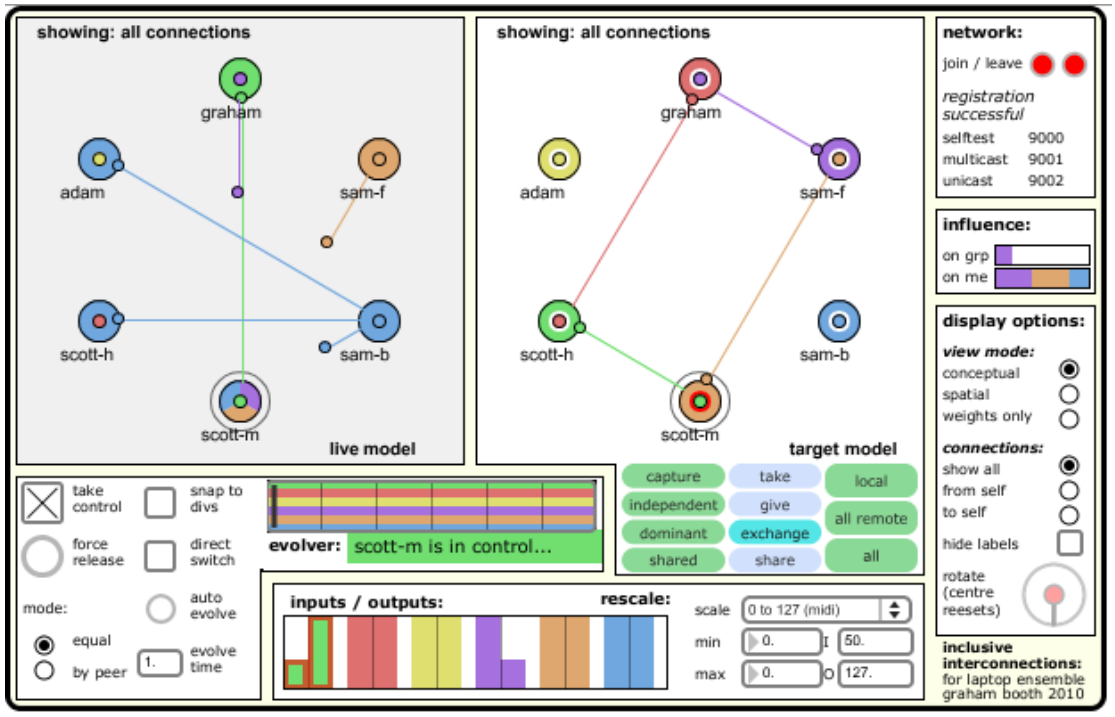


Figure 22: Presenting the Inclusive Interconnections user interface

Within this section, the key features of the Inclusive Interconnections system will be discussed in stages, in the same order that a player might encounter them on first use. Each subsection is covered as an additional screencast video demonstration provided in Appendix B[25]. In addition, a full user guide can be found in section iii. of Appendix A, which identifies the controls available and describes their key functions. These are referred to at key points throughout the text.

In order to begin using the system, players need to build or adapt a software instrument so as to supply one input to (and receive one output from) the Inclusive Interconnections application. This can be achieved within any Open Sound Control enabled application and is a core feature of the system's inclusive design. To further clarify this process, Figure 23 provides an overview of the system showing three players interfacing with the system each using different software and hardware. This demonstrates the flow of musical information through the system. This begins with each player's physical actions, which are then parameterised by the player's hardware and software, before being sent as Open Sound Control messages to the Inclusive Interconnections application. The dotted lines at the right hand side of the figure indicate data being routed across the network to all players, with the

[25] With the exception of the section on *Modifying/Building an Instrument.*

resulting modified outputs being passed back to the users application to contribute to the overall sound output. As can be seen in the example in Figure 23, Adam is the player currently in control of the live model of interconnection, thus enabling him to alter the parameter weightings globally across the network. Figure 24 then shows the updates that occur globally on all machines as result of a change in these parameter weightings or in any player's input value.

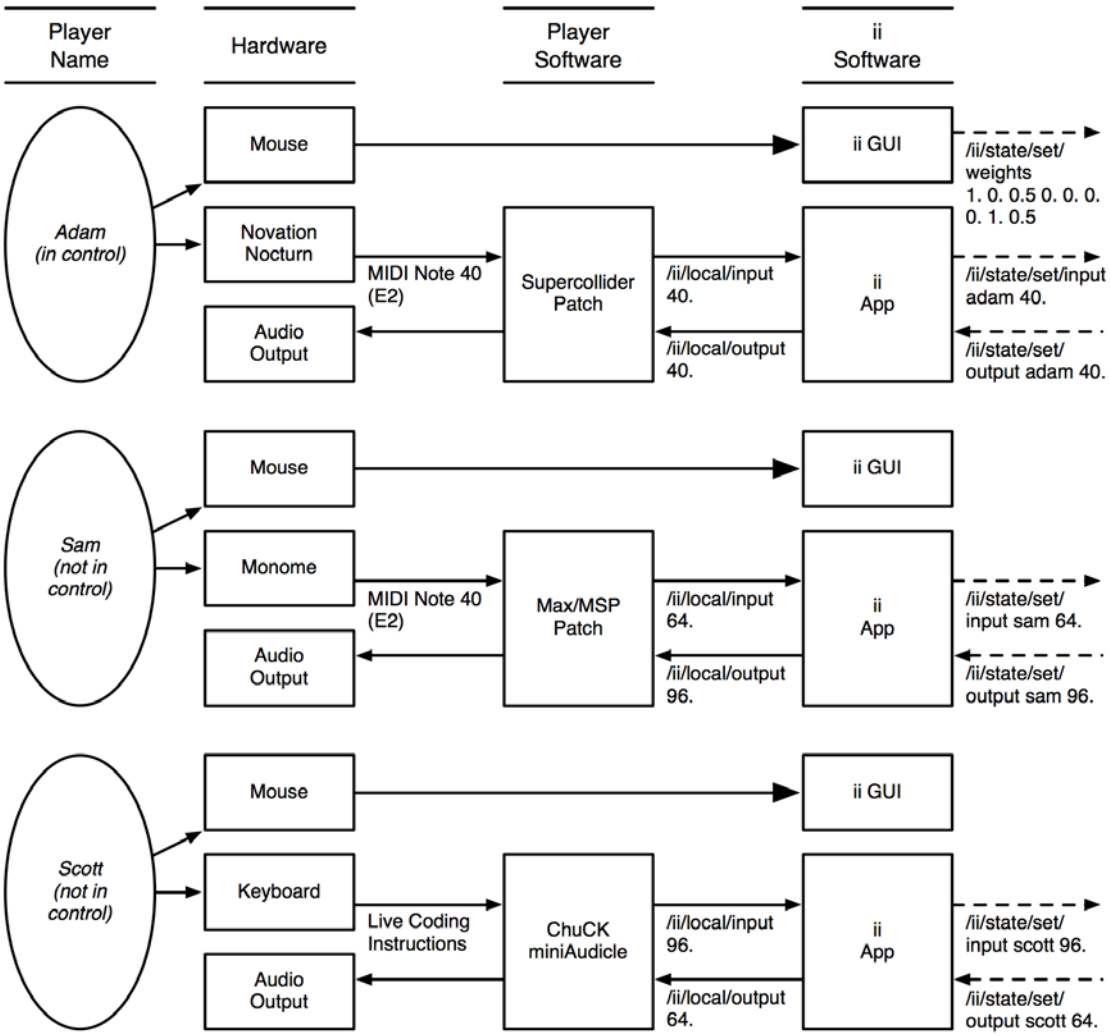


Figure 23: Overview of the Inclusive Interconnection system showing three players using a range of hardware and software

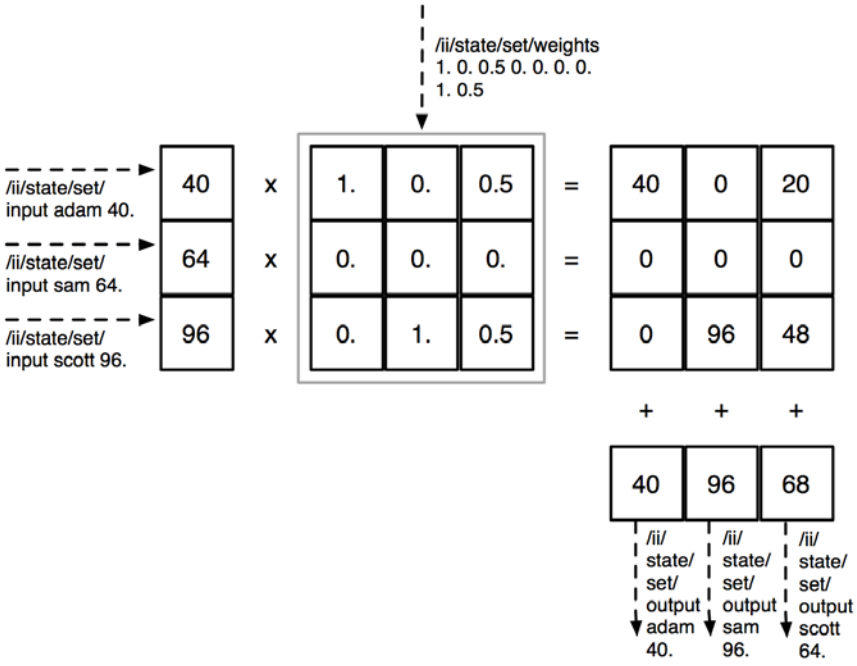


Figure 24: Overview of the global arrays stored on each player's machine

2.3.1 Getting Started: Modifying/Building an Instrument

The general procedure of modifying an instrument is detailed in Figure 25, which shows an example implementation in Max/MSP. For reference, this patch can also be found in section ii. of Appendix A (entitled *inclusive.interconnections.mypatch.maxpat*).

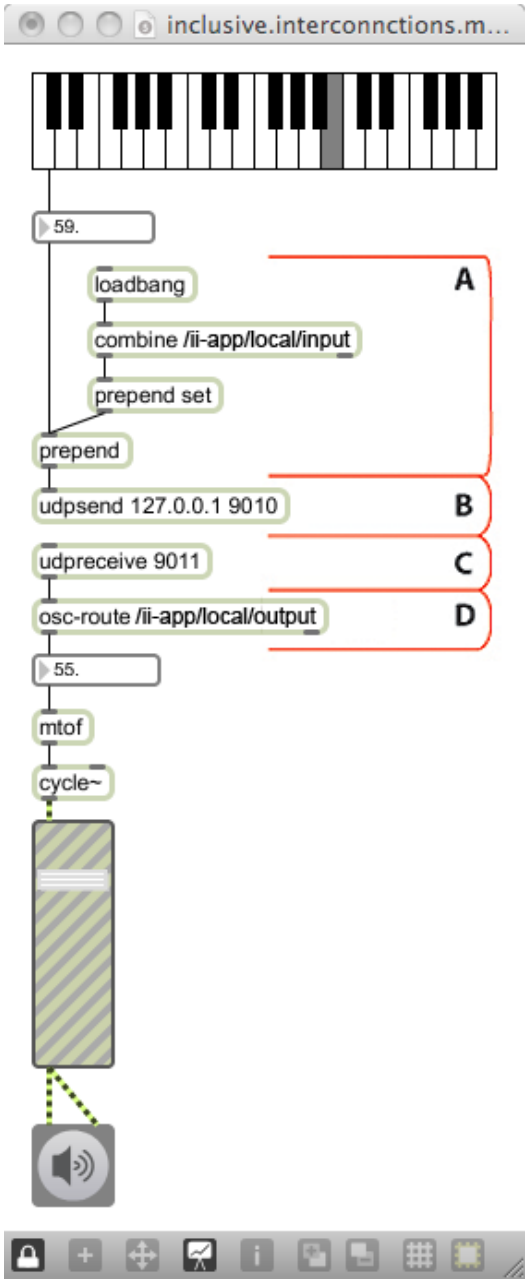


Figure 25: Modifying an instrument for use with the Inclusive Interconnections application (Max/MSP example)

To begin with, players must divert their input from its existing destination and add the following Open Sound Control namespace in front of the parameter value (see Figure 25, label A):

```
/ii-app/local/input [parameter value]
```

These incoming communications take place on port 9010 (see Figure 25, label B). For the output, the opposite applies, with the instrument receiving information locally from the Inclusive Interconnection application on port 9011 (see Figure 25, label C). This input is then filtered using the following OSC namespace (see Figure 25, label D):

```
/ii-app/local/output
```

The result is then reconnected to the original output destination.

2.3.2 Joining and Leaving the Network

The registration process is handled by the *network* section of the patch (see Figure 26). Registering is a simple matter of clicking the join button (see Figure 26, label a) and typing a name in the dialogue box that appears. Feedback on the status of the registration is provided below (see Figure 26, label c). If successful, the registration button is disabled (turns red) and de-registration is enabled.

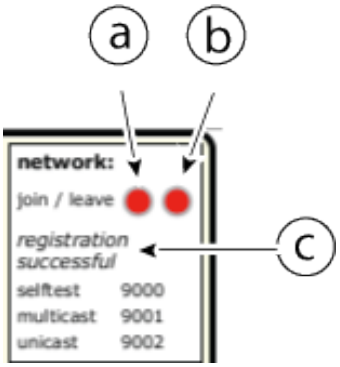


Figure 26: The *network* section of the Inclusive Interconnections user interface

By design, a player's instrument will operate in bypass mode until they have successfully joined the network. This allows the instrument to be played as normal if the player decides they want to step out of the parameter sharing process for a while.

Leaving the network is a slightly more complex process due to the need for the player to first remove any influence they have over the rest of the group[26]. Once this has been done, the de-registration button turns white, allowing the player to leave the network.

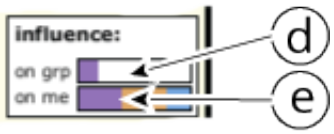


Figure 27: The *influence* section of the Inclusive Interconnections user interface

Figure 27 shows the *influence* section, which was introduced to aid deregistration, by showing how much control the local player has over members of the group (see Figure 27, label d). An additional bar has also been added to show the amount of influence that the members of the group have over the local player (see Figure 27, label e). This is also useful in a wider performance context, for example when there is no time to consider the full complexity of a model of interconnection.

For a full demonstration of the process of joining and leaving the network, see the accompanying network demonstration video in section i. of Appendix B.

2.3.3 Live and Target Model Displays

In order to use the system effectively, a basic understanding is required of how the interconnections between players are represented on screen. This is handled by the *live model display* (see Figure 28, label f) and the *target model display* (see Figure 30, label k), with the former showing the currently active model and the latter showing an ideal model. This difference in usage is reflected by the fact that the live model is always the same for all group members, whereas the target model is unique to each player. The majority of elements are common to both displays and have been designed with direct visual comparison in mind.

[26] A more detailed explanation of creating target models is provided in section 2.3.5, but for now, the easiest way for a player to remove their influence is to select all players in the target model display and choose *take* followed by *local*.

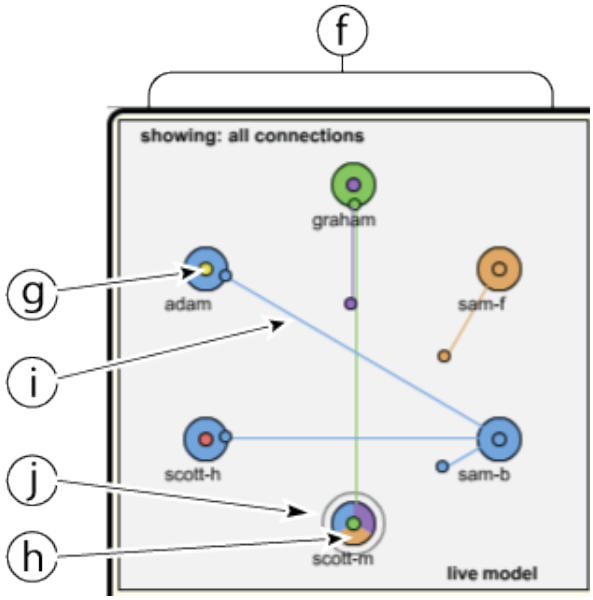


Figure 28: The *live model* section of the Inclusive Interconnections user interface

On both displays, registered players are represented by circles, which consist of an inner core and an outer ring, with the local player shown at the bottom and highlighted by a grey circle (see Figure 28, label j). The inner core (see Figure 28, label g) shows the local player's identifying colour, whilst the outer ring (see Figure 28, label h) shows who has control over the local player's shared parameter at any one time.

It could be, as when a player first joins, that the inner and outer ring will both show the same colour. This provides a clear identification of when a player is in control of their own parameter. When one or more players take control of the local player however, the ring is divided into segments that represent the amount of influence each player has (see Figure 28, label h).

As well as the outer rings, the influence of the local player on others is shown by outgoing lines in their identifying colour (see Figure 28, label i). In this representation, the length of the line shows the amount of influence the player has, with each line terminating in a small circle. These help to emphasise changes in the amount of influence held.

In addition to the elements described above, there are a number of additional visual differences to the target model, which reflect the fact that it can be manipulated by selecting players with the mouse. For example, the space between the inner core and the outer ring is used to show the selection status of each player (and turns black when

selected; see Figure 30, label l). Also, note that this space turns red for the local player's node, indicating that they cannot be the target of any transformations.

2.3.4 Display Options and Additional Features

The display options section (shown in Figure 29) provides a number of possibilities for adjusting both model displays according to the needs of the player at a given time. The first of these allows selection of three possible *view modes*. By default the application opens in *conceptual* mode, where players' nodes are equally spaced around a central point. In contrast, *spatial* mode allows players' positions to be moved in order to mimic their physical positions. The aim is to create a direct link between the interconnections seen on screen and the sound heard in the performance space. This can be seen as a form of 'control monitoring', to be performed in conjunction with listening to the output of different members of the group. In this mode, players may move their own node and those of others by dragging them with the mouse. The positions are then updated across the network on mouse release. By default, nodes are offset to presume a situation where players are facing each other, but the rotation control can be used to adapt to whichever direction a player is facing.

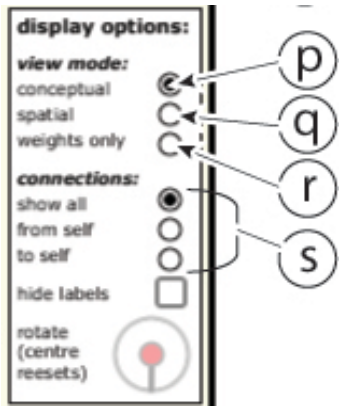


Figure 29: The *display options* section of the Inclusive Interconnections user interface

The latter *weights only* option (see Figure 29, label r) displays a bar graph of influence levels, indexed by player colour. This provides an alternative way of visualising the amount of influence each player has and over whom. Finally, the *connections* section (see Figure 30, label s) allows connections between players to be hidden, in order to emphasise the

influence on, or influence from, the local player. These are applied using the *from self* and *to self* options, respectively.

2.3.5 Setting Up a Target Model

As we have seen previously, new models of interconnection are created by performing operations on the existing target model. The left-hand set of buttons below the target display (see Figure 30, label m) provide a number of starting points, including the previously discussed independent, dominant and shared archetypes. In addition to these, a capture mode is provided, which allows players to take a snapshot of the current live model in order to subject it to further transformation. This makes it possible to create continuously evolving scenarios.

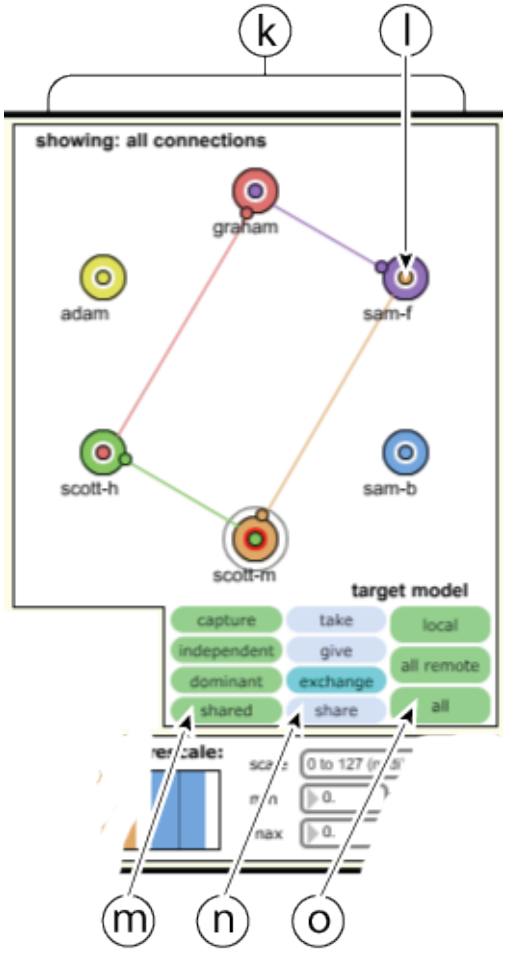


Figure 30: The *target model* section of the Inclusive Interconnections user interface

For individually tailored transformations, players must select one or more of their peers on the target display, either by clicking on their cores one-by-one or by dragging a box around

multiple nodes. Once one or more target peers have been chosen, the local player must choose a type of transformation operation and decide whose influence will be involved in the transfer itself. This is achieved using the second and third set of buttons under the target model (see Figure 30, labels n and o). It is worth noting here that only the third set of buttons (i.e. *local*, *all remote* or *all*) actually apply the transformation. For a complete description of the different operation types, please refer back to section 2.1.5.

2.3.6 Applying a Target Model

The purpose of the *evolver* section is to allow players to impose the target model they have created onto the current live model, in order to apply it to the current performance situation (see Figure 31). To do this, a player must first click the *take control* switch (see Figure 31, label t), at which point their identifying colour and name will appear under the *evolver bar*. As only one player may take control at any given time, this provides a useful indication of who is currently in control. In addition, the *force release* button allows players who are not in control to interrupt the controlling player whilst they are in the process of applying a model, for example if they have had control for too long, or have crashed.

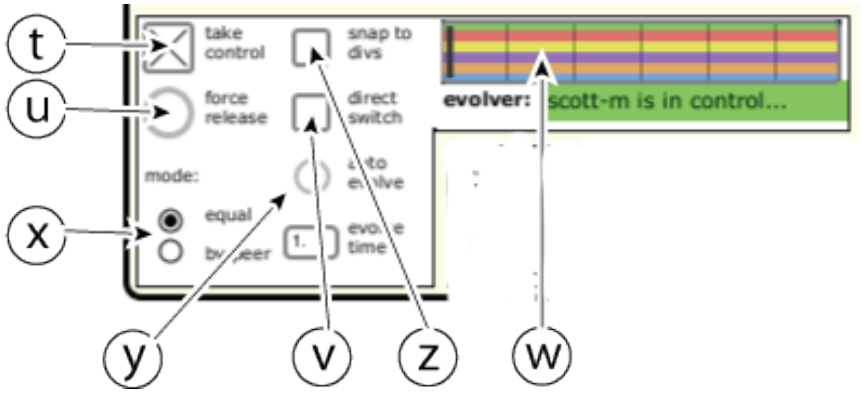


Figure 31: The *evolver* section of the Inclusive Interconnections user interface

Once control has been taken, it is possible to move between the live and target model in a number of ways. The first of these is to use the *direct switch* to immediately apply the target model to the live model (see Figure 31, label v). Depending on the choice of parameter shared, and the design of each player's individual instrument, this can cause jumps between the old and new output values, which may or may not be desirable.

An alternative option is to use the *evolver slider* to scrub between the two models in real time (see Figure 31, label w). Using this approach, control may be released at any point to create a hybrid model, whilst the *mode* setting defines the order in which influence will be

exchanged when interpolating between the two models (see Figure 30, label x). In *equal* mode, transitions of influence between players occur concurrently until the transfer is complete. In *by peer* mode, however, the transfer of influence takes place sequentially, on a player-by-player basis, which can lead to more useful models at midpoints along the way. This difference in mode is reflected in the appearance of the evolver slider, which displays either horizontal or vertical colour bars. In addition, the *snap to divs* button (see Figure 30, label z) may be used here to create more discrete 'stepped' transitions.

The third and final option is to make use of the auto-evolve section to generate smoother automatic transitions between the two models (see Figure 30, label y).

3. Testing, Evaluation and Results

With an initial implementation complete, focus shifted to a one-month period of intensive practical testing and evaluation, which centred on weekly two-hour sessions held with members of the postgraduate Huddersfield Experimental Laptop Orchestra (H.E.L.O.pg)[27]. Both the testing and evaluation sessions required that users had some previous experience of creating and performing with their own software instruments, in order to evaluate whether or not the system was compatible with a range of working methods. In this sense, the members of H.E.L.O.pg can be seen as an appropriate group of experts in terms of both the aims and scope of the project.

3.1 Testing

3.1.1 Overview of Testing Sessions

In total, three testing sessions were conducted, which each consisting of a series of small-scale tests in limited circumstances. The aim of each session was to improve the overall stability of the application and to conduct an early evaluation of the interactions taking between players. During this time, a number of core improvements were made to the system and new features added based on feedback gained, which are detailed in the following sections. These sessions also served to familiarise players with the basics of the system, in order to foster a deeper understanding of its use by the time of the evaluation.

3.1.2 Aims and Results of Testing Sessions

3.1.2.1 Session 1

The first session focussed on testing and improving the underlying networking aspects of the system as enabled by early versions of the *net.table*, *net.ip.local.test* and *net.transport.peer* abstractions. These had been tested previously in a computer lab environment, but not in a situation comparable to real world usage.

[27] The pool of players drawn on during these session were Sam Birkhead, Sam Freeman, Scott Hewitt, Adam Jansch and Scott McLaughlin.

The specific aims of the session were:

a) to evaluate detection and testing of users network adapters (as handled by the *net.ip.local.test* abstraction),
b) to see if an identical table of registration data was being reliably maintained on each machine, regardless of the order of registration,
c) to evaluate unicast delivery of data to all peers on the network using the peer transport (as handled by the *net.transport.peer* abstraction),
d) to test varying combinations of wired and wireless adapters on the network.

The testing group consisted of the author, Sam Freeman and Sam Birkhead, running two 13" Apple Macbook Pro's and one Samsung netbook, respectively. Networking was handled by a TP-Link TL-WR941ND router. Whilst not a large-scale test or particularly diverse, using a limited amount of machines reduced the number of possible configurations, making it simpler to do exhaustive tests in order to track and reproduce problems.

For testing purposes, players ran the *inclusive.interconnections.table.test* abstraction (see section ii. of Appendix A), which allowed them to register and deregister on the network, whilst seeing the names, IPs and registration order of each player. In addition, an early version of the *net.peer.transport* abstraction allowed each user to move an on-screen slider, which affected their own slider as well as those of all other registered players. This system was used to test the overall responsiveness of the network. Another section of the patch allowed the time grain of a ramping control signal to be adjusted, which was aimed at testing how quickly values could be sent over a wireless network before reaching a bottleneck. For these tests, laptops were placed in line in order to visually evaluate their responsiveness.

In terms of a), the *net.local.ip.test* patch had no problem automatically selecting the fastest available adapter. However, some sluggishness was noted in terms of the total time taken for players to register on the network. In direct response to this the system was overhauled to reduce the number of steps required to register to two, bringing the current system in line with the registration process described in Section 2.2.

In regards to b), correct registration was observed on all machines, except in situations where the first player deregistered and another registered in their place. This led to the existing indexing system being swapped for a stack-based approach as described in Section 2.2.2.4. In evaluation of c), it was found that all registered users were able to take control of

the shared slider across all machines, showing successful and timely unicast delivery of messages using the peer transport. Finally, for d), it was suprising to find that wired and wireless networks proved comparable in terms of slider responsiveness. When tested with the time grain ramp however, the netbook was found to receive values slightly later than the two other machines. This only proved to be problem with transmissions at intervals of less than ten milliseconds, with responsiveness remaining usable above this threshold. In anticipation of this delay increasing with larger scale networks, this limit was doubled to place the transmission threshold at a maximum of one message every twenty milliseconds (i.e. 50Hz). Whilst not taking into account any additional delay, or varying network jitter[28], this was deemed more than acceptable for control signals, but as a result it is recommended that players use some degree of interpolation within their individual instrument designs.

3.1.2.2 Sessions 2 and 3

The subsequent testing sessions introduced members of the group to the full *Inclusive Interconnections* application. The second session took place with four players[29] and marked the first real-world test of the full system. At this stage, players were provided with a basic instrument patch in Max/MSP which had been pre-configured to share pitch values with the rest of the group (this patch was identical to the one shown previously in Figure 25). At this stage, players were asked to try all the available features of the system, without any prior explanation, in order to discover bugs and expose limitations.

The third and final testing session took place with a full cohort of all six players and took the form of a workshop where players developed their own instruments by either adapting the previously provided test instrument in Max/MSP or by using their own approach. During this session, a full test of the system was captured on video and can be seen found in section iii. of Appendix B. This is provided both as a live performance video and as directly comparable screencasts taken from four of the six players.

Due to the limited nature of this early version, players generally stuck to applying preset archetypal modes[30] rather than creating their own. The issues of greatest concern at this stage were the appearance of empty nodes on screen at registration and the appearance of nodes with a combined influence that did not add up to that of a whole player. These issues

[28] Forward synchronisation issues have been noted, but remain beyond the scope of this project.

[29] Adding Scott Hewitt on a 2Ghz Macbook to the previous line-up.

[30] Including a random mode which was present at this stage.

were pinpointed to a combination of re-registration issues and later joining players not picking up the influence of existing players in some cases. This was also compounded by the fact that users were able to leave the network whilst exerting influence over each other, causing this influence to be lost. Following on from the session, a number of approaches were put in place to deal with these issues, including a system that only allows players to deregister when their influence over others is zero. This went hand-in-hand with the introduction of influence bars as part of the G.U.I., in order to aid identification of personal and group influence. Whilst in themselves these measures do not make any allowance for system crashes, it was thought more important to focus on ensuring the stability of the existing implementation rather than attempting to address this complex problem at this time.

Encouragingly, despite these technical issues, the session highlighted the overall potential of the application in terms of its ability to enhance social interactions between players. Players proved to be engaged by the interface throughout and found novel ways to emphasise their interconnection with others by adapting the designs of their instruments.

3.2 Evaluation

3.2.1 Overview of Evaluation Session

The final evaluation consisted of a practical period of rehearsal using the system, directly followed by a focus group session. Four members of the same expert group were drawn on here, as in the previous technical testing phase[31]. This was in preference to conducting a blind test with new users, as it was thought that this would provide deeper and more valuable insights into the system's overall effectiveness. That the participants already knew each other and had interacted using the system in the previous technical studies was seen as a significant benefit and in these terms the group can be regarded as homogeneous and qualified to comment.

[31] Specifically, Sam Birkhead, Sam Freeman, Scott Hewitt and Adam Jansch.

3.2.2 Focus Group Methodology

The overall aim of the focus group session was to gather qualitative data on the inclusivity and mutability of the system's operation (as identified as core aims in Section 1.4), as well as to identify areas for future development. As a research method, focus groups are characterised by moderated discussion and can be said to combine elements of other qualitative methodologies, such as individual interviews and participant observation, whilst at the same time providing access to data that these methods cannot (Morgan 1988).

For the purposes of this project, it was particularly important to select a method which could reveal how members of the group interacted with each other when using the Inclusive Interconnections system. Therefore, whilst individual interviews are desirable for their ability to elicit a focussed and comparable set of responses from each participant, they are unsuitable here as they do not allow for direct interaction between participants (King and Horrocks 2010). Instead, it can said that in order to gather data regarding the operation of what is essentially a social system, a methodology is required which is able to take a more direct account of social behaviour. The technique of participant observation meets this aim by collecting data from naturally occurring social settings. Using this method, groups are studied in situ, without the potentially negative influence of a moderator. The problem with employing this approach for the purposes described here is that no natural setting for the discussion of parameter-sharing systems exists. Beyond this initial problem, it can also be said that a free-flowing discussion would be unlikely to focus clearly enough on the specific research problem at hand.

Alternatively, by marrying the directive approach of individual interviews with the less formal social setting of participant observation (Morgan 1998), focus group methodology has the potential to address some of deficiencies outlined above. Using this approach, the key topics for discussion are supplied by a researcher, but the group setting allows for unstructured interactions between participants to take place and emergent issues to arise. This combination was considered well matched to trying to understand the benefits of social interaction in the context of individual experience, as when using the Inclusive Interconnections system.

3.2.3 Focus Group Implementation

The focus group session was moderated by the author, with participants being asked a series of open-ended questions regarding their practical use of the Inclusive Interconnections system. The timely nature of the discussion made it possible to capture responses from all users while their experiences of using the system were still fresh in their minds. The majority of questions focussed directly on the core aims of inclusivity and mutability, as identified in Section 1.4 (e.g. *'When manipulating the target model, how clearly did you understand the kinds of interconnections that would result?').* These questions were generally asked of the whole group, but occasionally specific players were asked to comment or elaborate on their responses in the context of their own working practices. This was done in order to draw out some of the finer details regarding use of the system. A smaller set of questions were more general in nature, inviting comments on aspects such as the overall interface design (e.g. *'Could you comment generally on the overall interface design, in terms of clarity and ease of use?'*) or to detail any bugs encountered or improvements considered during the session. These open-ended questions left enough breathing space for issues to arise naturally and often led into wider discussion of future uses for the system.

The practical component of the session was far more rigidly defined than in the previous testing sessions and consisted of an introductory demonstration, period of free-play and a structured task. The demonstration covered a number of changes which had been added since the last session and focussed specifically on the process of creating target models, which had recently been overhauled.[32] This also gave a chance for players to ask questions about any aspect of the systems operation and provided an opportunity to get up to speed for those who had missed previous sessions. The subsequent period of free play then allowed players to test out their understanding in a practical sense.

Following on from this, each player was asked to create a model of interaction which contained primarily dominant, shared or exchanged characteristics. This was done without the knowledge of the other players in the group. Once created, members of the group were asked to play without discussion, introducing their models at any time during the performance. It was hoped that, by asking players to generate a specific model, rather than playing freely as in previous sessions, that this would stimulate more focused feedback when it came to evaluating the overall mutability of the target model.

[32] This was similar in form to the video demonstration contained in section i. of Appendix C.

3.3 Results

This section presents selected comments and themes from the evaluation session, which have been organised in subcategories under the core aims of inclusivity and mutability. As far as possible, these are presented as is, reserving discussion of wider implications for the following section. An audio recording of the full discussion can be found in Appendix C.

3.3.1 Inclusivity

3.3.1.1 Comments relevant to aim 1a

(‘Ability to quickly and seamlessly share a parameter with the group').

Hewitt commented on ease of integration, as follows:

> What you've actually got is something which takes all of that information and turns it into a single numerical stream and that makes it […] fluid and very quick to work with. [...] I've played in a couple of systems where I've had a fully exposed thing and it shows me fifteen parameters being generated by fifteen players and in that thing it's overwhelming in terms of trying to use the information [...], whereas with the system you've got here, you give me a single value to work with but [...] while the value is arbitrary it can be made of a whole infinite myriad of possible sources…

3.3.1.2 Comments relevant to aim 1b

(‘Ability to integrate with players' existing software and hardware during performance').

Hewitt provides insight into using the ChuCK language to build responses to the system from scratch, in real-time, in a live-coding context:

> Essentially, obviously you're just supplying a piece of information and from building in a very modular way [...] it's just a matter of electing to have this piece of information do this role rather than some other role. In terms of design, I've limited it to driving a pitch of a system that was going on initially and then I set it up so that […] it stepped through the information at a much slower rate than the rate to which it was receiving the information, so […] when somebody did a kind of quick gesture it slowed their gesture down to apply it over a couple of seconds rather than instantaneously...

Birkhead highlighted two issues from the perspective of using a netbook computer with the system. Firstly, regarding the amount of system resources consumed, it was noted that *"it's taking up about 50% of my CPU just running the interface, so there's not that much more for instrument building."* The second issue highlighted the implications of having a reduced amount of screen space to work with: *"I've been using the spaces inbuilt in OSX to flip*

between the interface and the instrument. [...] so [I] hold control and go left and right to flip between the two."

In terms of on-screen integration for other users, all found it possible to display both their own and the Inclusive Interconnections user interface at the same time. Freeman commented: *"That's okay for me. I often divide my screen space so I'm only using half of it for one thing or a quarter of it for another thing, that's something I'm used to. [...] If you'd have made a full screen thing that would have been a problem, but because you've limited it…"*

These screen space issues led into a wider discussion regarding attention, including whether players are able to continuously monitor the current state of interconnection. Hewitt commented: *"If the models are constantly shifting and changing then [...] in my mind it demands a lot attention to follow [...] and that kind of undermines the performance beyond the interface. [...] For me, that would defeat a lot of my intention because there's no way that I could monitor in real time and work out the consequences of those continuous real time changes."*

When asked about use of the system alongside a sequencing package, Birkhead commented, *"I don't necessarily think Renoise is an environment for it. I mean you could trigger off loops or whatever but I don't really think it would be...I think Max is the perfect environment for this...something more direct and interactive."*

3.3.1.3 Comments relevant to aim 1c

('Ability to join and leave a performance at any time').

Although the ability to join and leave the network mid-performance could not be tested due to registration issues with the patch during the session, it was still considered beneficial to ask users to comment on this issue, in terms of the value this might add to the system in future.

Discussion pointed to a number of benefits for improvised or composed settings. In terms of the former, Jansch stated one possible benefit as being*: "If it goes all awry you can jump out of it quickly. If things are going a bit crazy you can just mute it quickly and then figure out what to do."* Freeman suggested a similar requirement in a live-coding context *"...you might want to say right 'just stop sending me data while for a minute while I change this, OK I'll accept the data again now'".*

In a situation where pre-determined models of interconnection are in use, Jansch also stated that it might useful if *"...in a certain section you weren't meant to play, you were just taken out of that model."* In contrast, Hewitt focussed on the need for reliability, stating that *"the main reason for [...] there being value in [...] people being able to join and leave at any time is the fact that it's inherently going to be resilient against things going wrong."*

3.3.1.4 Comments relevant to aim 1d

('Potential for flexible use in a range of open and closed performance situations (e.g. performer/composer led)')

Even with limited use within a relatively short time frame, players were able to assess the flexibility of the system and suggest a number of ways in which it could be streamlined or extended in order to achieve specific performance goals.

One feature request was the ability to recall previously stored models of interconnection, or to go off on branches from models players found particularly usable. Hewitt commented: *"It [...] feels like I can build really complex things [...] and if they're complicated then they probably do take a while to build, but the fact that [...] they [...] get thrown away and I never see them again is the problem."* Hewitt also proposed a composition-oriented version of the system based on being able to store presets, as follows*: "I could see very effectively how I could have three or four pre-built models and use them as sectional structures and I could see how I would be able to learn those models and how they work and then be able to perform my user interface to play that."* Hewitt goes on to add that *"...I know that after one or two rehearsals I would be able to fully understand what the consequences of each model [are]…"*

In support of such an approach, the group proposed the addition of a system whereby 'dummy' players are represented on screen at the outset, even if no decision has yet been made as to who will control them. Jansch stated that: *"...to build a preset system you have a have a dummy system because [...] it'd be impossible to open a preset without all the people there, all connected"*. Jansch also went on to say, *"...you could maybe grey out, do some kind of user interface thing that says this person has decided not to be controlled…"*

In contrast to composer-focussed approaches, other players considered how they used the system individually in performance, and how this might be extended. In terms of performance control, Birkhead identified the input/output bars as providing a better indicator of end result, as opposed to the weightings shown in the model of interconnection: *"...from the inputs and outputs you can see if I move something what effect that has on other*

people, and I found that when there was a shared model and there was something complicated going that was the easiest way to figure out what was going on." As an extension of these concerns, Hewitt suggested that *"...if there [were] two more graphs and one of which showed […] the consequences of me inputting some information in terms of does it go to other people and then another one which showed other people moving stuff [...] that would probably clarify what was going on a lot more."*

Further discussion revealed that there remain fundamental questions about choice and control when using the current system in an improvised context. Birkhead stated: *"I think something that's really important when you're improvising or when playing live...you've got to know what your actions are. [...] It's quite hard sometimes with the system to know exactly what's going to happen. You don't know whether or not to move your slider at this time, because it will make an awful mess, or whether or not it's going to make something amazing...you don't know all the time and I think the uncertainty of that is a bit..."* (Jansch picks up the thread) *"...it's a big unknown..."*

Jansch elaborated on this point further, emphasising that, in an improvised setting, there is a need for each player to be able to define the terms of the interaction for themselves, rather than having it decided for them by other players: *"Say I'm generating some data, [...] I can make that parameter or a number of parameters available […] to other people to integrate into their workflow, but no one can just take over my system".* Using such a system, players would *"...register whichever parameters they want to register and [...] then I could just drag a cord to somewhere which would then say, okay connect that to my 'port one' and then that [..] is connected into whatever thing I have running in another patch and then you can freely connect them as you see fit."* This points to one way in which the system might be repurposed for use in an improvised performance setting.

3.3.2 Mutability

3.3.2.1 Comments relevant to aim 2a

('Visually understand the current model of interconnection that applies to the group').

During the discussion, a number of comments pointed towards improvements that could be made to the visual representation of the model of interconnection. To begin with, the group were asked to comment on whether they found the graphical models easy or difficult to understand. Birkhead responded by saying: *"When there's a simple model in the target view*

it's quite easy [...] when someone is dominant it's obvious what's going on and you can tell that [...] but when there's something complicated going on in the system it's a lot more difficult."

In terms of representation, some players preferred one type of visual representation to another. For example, Birkhead seemed to prefer the *weights only* view, voicing a desire to be able to manipulate this model, stating: *"most of us have been staying in the lines diagram because you can't set up a model easily in the histogram."*

Others stated that the visual representation proved more useful in understanding the influence they had on others, as opposed to the amount of influence others had on them. Freeman stated: *"The lines in-between them make sense to me, and I have a vague idea of what's going with the colours pie chart type display, but the lines make more sense to me than the pie chart."*

Jansch identified a possible issue with the directional representation of the lines: *"I think maybe the lines go the wrong way, I'm not sure. […] Maybe the problem is thinking of it in taking and giving isn't the right way, maybe it's better to think...say I'd like to get Scott's parameter so drag a cable from Scott to you, or towards you, or I'd like to give my parameter to Scott...do it the other way...so rather than I think maybe it's very similar, but I think there's a slight conceptual difference there."*

3.3.2.2 Comments relevant to aim 2b

(‘Manipulate the model to achieve a wide range of possible forms of interconnection').

The improvement requested most by players during the discussion was the ability to be able to manipulate the target model in a more direct way than was currently possible. Hewitt stated: *"If I could grab the end of the line [...] and just like [...] pull it towards a certain direction I'd would be able to do that [...] and know exactly what was going on...but [...] if I wanted to extend Graham's influence over myself slightly...then that...I'm not entirely...obviously, I would capture the live model and then I would kind of highlight the two of us and kind of hit 'give' a couple of times [...] and I think that would do the job but I'm not sure."* Jansch added: *"If you were to be able to grab cords and connect them to people or to move them towards other people, you get more of an idea of what you're doing...whether you're giving your parameter away or if you're going to drag one the other way."*

Freeman commented on the sequential nature of applying operations: *"...it's like a Rubik's cube isn't it. If you want to go from there to this thing I imagine I want, to this, then that, then that, then bring this back and swap that over there and now I've got it."* In addition, when asked how easy it was to understand the model he had created, Freeman responded: *"I'd say, sixty to eighty percent of the time I know what's going on, but the rest of the time I press the button and it doesn't do what I thought it was going to do."*

3.4 Outcomes and Directions for Future Work

In this section, the results of the previous section are discussed in order to highlight the successful aspects of the project as well as to identify areas which remain to be addressed in future work.

3.4.1 Extending Inclusivity

3.4.1.1 Integration

The Inclusive Interconnections system can be considered successful in achieving integrated operation within the context of the different approaches used by members of the testing group. During this period, it was noted that all players, regardless of software used, were able to adapt their existing instruments quickly and easily in order to begin sharing a parameter over the network. At present this can be regarded as one of the strongest aspects of the system's design, leveraging both Open Sound Control and local UDP communication for seamless integration. In addition, the added simplicity of a one-in, one-out approach to parameter-sharing provided an incentive for users to experiment with the system, presenting a low barrier to entry whilst still retaining the richness provided by a mutable model of interconnection.

In future, testing with a broader range of approaches is required in order to further extend inclusivity. Future versions of the system will seek to look beyond OSC in order to support traditional sequencing environments via MIDI integration. Whilst it is certainly true (as Birkhead comments in the results) that developing bespoke performance tools in direct response to the affordances of the system would be preferable, it remains likely that some members will still favour an off-the-shelf approach as a starting point. I include in this category members of the undergraduate laptop ensemble at Huddersfield, who primarily use environments such as Logic, Ableton Live, Reason etc. With this in mind, there would be significant advantages in terms being able to host existing virtual instruments, or build

performance systems from pre-formed components. Future support for this could be provided by integrating MIDI input and output capabilities into the Inclusive Interconnections application itself, which would a relatively straightforward process.

3.4.1.2 Towards a Lightweight, Modular Approach

On a purely technical level, there remains a longer term need to redesign the system for lightweight, cross-platform operation, which is able to support both Linux users and those using less powerful machines such as netbooks. Two possible responses to this have been identified, both of which suggest moving towards a more modular design, as suggested by Birkhead during evaluation.

The first of these would be a centralised, modular system consisting of separate network-handling, weight calculation and user interface components. It is envisaged that the first and last of these could each be written in a cross-platform language, with the underlying weights calculations being performed remotely by a centralised server. This kind of separation is implicit in the existing design, with code being split into *net.**, *vis.** and *mut.** components (for a glossary of these existing modules, see Appendix A, Section iv.). Whilst a decentralised design is not ideal for reasons identified earlier in Section 6.1, these disadvantages would be outweighed by the benefits to inclusivity that a lightweight, cross-platform design provides.

A second option would be to rewrite the current decentralised, single-application system in a lower-level language, such as has been achieved within the Bridges project (Wyse and Mitani 2009). Again, this could be approached in a modular way. This remains the ideal solution, but would require significant additional development time in terms of optimising the code for netbooks and in rewriting the existing code for enabling matrix calculations. Regardless of the option chosen, further research is required in order to identify suitable cross-platform development environments in which this work might be conducted.

3.4.1.3 Screen Space and Attention

The results of the evaluation also highlight a number of problems related to the dual-application architecture of the system, in terms of both screen space and attention. For example, whilst having both the Inclusive Interconnections and user instrument interfaces open on-screen is necessary, this is not currently possible for netbook users. On one hand, Birkhead was able to quickly draw on his existing performance experience with a netbook to come up with a workable situation that involved switching between screen spaces. On the

other however, this meant that it was not possible to see the current model of interconnection at the same time as performing with an instrument patch. This was a significant issue, given that the live model may change at any time.

Even for the majority of users who could see both interfaces concurrently, the implications of having both on-screen caused some difficulties in terms of being able to monitor the complex interactions taking place. Whilst in one sense this is a case for exercising restraint in the use of the system, it may also strengthen the case for developing separate performance and editing interfaces, as covered in the subsequent section on flexibility.

3.4.1.4 Joining and Leaving

In line with Freeman or Jansch's comments in Section 3.3.1.4, it can be said that a truly open approach to parameter-sharing means giving players the power to drop out of an interdependent situation when it doesn't meet their requirements or expectations. Such an approach is key to a player's sense of his or her own performance identity. So far, a number of steps have been made towards a system that affords inclusivity in terms of players being able to join and leave the system at any time. These include provision for stateful operation (where newly registered users pick up the current live model of interconnection on joining) as well as the implementation of a mechanism for enabling players to leave 'without a trace' (in terms of their influence on other peers).

At this stage, the first priority is to continue to work to address issues in the registration process which have yet to be solved reliably in real world situations. Following this, it is intended to implement a 'keep alive' system, as discussed during initial testing, whereby users' presence on the network is continually monitored, such that if they leave (whether by closing or crashing) their influence over other players will be automatically restored to each source player. Whilst closing the patch mid-performance might prove an impolite way to leave, and cause a jump in players input and output values, it would also ensure complete reliability in a performance context, as well as addressing the wider issue of inclusivity in terms of players being able to leave the performance at any time.

3.4.1.5 Flexibility

It was clear from the feedback provided by the testing group that a key aim of future work should be to extend the existing system to further support both performer and composer oriented uses of the system. These concerns approximate the score-led or improvisation-led

approaches implicit in Winkler's performance models (1998), as discussed in Section 1.2.4. These are not mutually exclusive areas, but rather emphasise differences in focus that the system should aim to support. For example, in the current implementation, any performer may 'compose' their own interdependencies, but these cannot currently be re-used later or in subsequent performances. This was highlighted by members of the testing group, who proposed that the system should support the creation of repeatable models of interconnection, which may be stored, recalled and moved through, either by means of a timeline or activated by real-time events.

Whilst on the surface, adding the functionality to save models of interconnection seems straightforward enough, on deeper investigation it becomes obvious that this presents a major conceptual challenge to the representation of players in the system as it stands. At the very least, recalling a previously saved model relies on knowing how many players will be available within that model, and some definition of whom these players will be in terms of their individual roles or voices. To address this in future, it is possible to envisage a 'dummy system' (as proposed by players in the evaluation session), wherein a fixed number of nodes are defined at the start of play, but which only become active when claimed by registering players. In addition, players could choose to opt into or out of a given model, so that even if not all players represented in the model were agreeing to participate at any one time, their nodes would still function as basic containers of influence. The key challenge of this approach, as has been identified previously, is to how to manage latent influence that is not controlled by any one player. As such the exact implementation of a preset system requires further consideration. However, the kind of repeatability that this kind of composition-orientated mode would require is certainly a strong test of the resilience of a interdependent system and would prove a significant milestone if reached.

In contrast to the above concerns, a performer-led approach requires a greater understanding of the moment-to-moment needs of each player. It was noted in the feedback that a different set of skills were required by players when a) performing with an interconnected instrument and b) manipulating the model of interconnection. These suggestions point towards the need for a performance view, where players are able to see a cut down version of the interface which shows only the currently applied model of interconnection. Such an interface should also be optimised in order to emphasise only the information that will help a player to understand the direct result of their own actions at any one time, thus allowing them to better manage the richness of the system in a performance

context. One example of this is Hewitt and Birkhead's suggested change to the G.U.I. which would place their input value directly alongside the corresponding output values they are affecting. This is one change that would go some way to enabling them to more clearly identify the potential effect of their own actions.

In the longer term, there exist far greater challenges in trying to making the system more performer-friendly. In terms of identity, Jansch's comments in particular point to the need for a system which enables creative interconnections to be formed under the control of the individual user, as opposed to the push-pull nature of group interconnection. Such an approach has the benefit of not intruding upon users individual musical voices without their express permission, and as such may be more appropriate in an improvised context.

3.4.2 Making Mutable Models More Intuitive

3.4.2.1 Improvements to Conceptual Representation

Freeman's comments in Section 3.3.2.1 would seem to confirm the initial findings at the design stage, that lines between nodes are more intuitive in terms of representing influence than segments are, despite the fact that they are often less directly comparable (e.g. if nodes aren't equally spaced). However, this could also be because, in exercising control over the voices of others, player's outward influence assumes more importance than others' inward influence on them.

In response to Jansch's comments regarding line direction, a proposed alternative representation could involve users pulling lines towards themselves to gain control, rather than pushing their influence outwards to others to control them. This would also seem to resonate with the idea of influence itself lying in a central place and may be more intuitive than trying to contain influence within an existing sounding body.

As well as pointing towards a performer-centred representation, Jansch's responses also emphasise the benefit of being able to manipulate the model in terms of users gaining an epistemological understanding of how the system works. In other words, the easier it is for players to manipulate the model, the more intuitive the overall graphical representation becomes. This can be seen as a reciprocal process whereby players test their assumptions about the graphical model itself by manipulating it to create new models. With this in mind, one argument for direct manipulation of the model is that this in itself provides a key to it being more easily understood.

3.4.2.2 From Operational to Direct Manipulation

During evaluation, players highlighted some concerns with the operational (or 'Rubik's cube') approach to generating target models of interconnection. Comments showed that users would prefer a more hands on interface, in terms of being able drag-and-drop connections, in order to adjust the balance of influence between individual players. Whilst this would be trivial enough to implement, there is questionable value in making it the core mode of adjustment. For example, creating an equally shared model would involve clicking and dragging each terminating circle individually, which in larger groups would be impractical. Therefore, whilst it is proposed to further develop the interface to add drag and drop capabilities, future efforts require identification of some higher level parameters, which would facilitate selection and manipulation of multiple weights in a way which remains intuitive.

3.4.2.3 Alternative Methods of Sharing Control

In terms of being able to understand the conceptual representation and aural result of a given model of interconnection, players to seemed to be quickly able to make sense of models with largely dominant, or exchange characteristics, but found shared models much harder to grasp. This was to be expected, as it can be said that the very nature of sharing control of a single parameter is a counter-intuitive process, where the influence of each player becomes diluted and it becomes more difficult to tell who is in control of who. With this in mind, it is debatable as to whether shared models should be represented more clearly or whether a complex mix of player's inputs should equate to a complex representation and sonic result. For now, it can be said that a wider study is needed in order to investigate how players might share control of parameters, particularly in regards to whether the process of sharing should attempt to encode individual meaning within interactions (such that they still prove relevant in shared situation) or whether the idea of sharing itself has to involve some compromise in terms of diluted influence.

3.4.2.4 Understanding the Implications of Interdependence

Finally, in tandem with development of the system, there is also a need for players to gain a deeper practical understanding of what it means to share parameters with others, and how to make use of interdependencies during performance. For example, greater consideration is needed of whom might take control the system at any one time, and under what conditions this might change. This necessitates some kind of hierarchical or social

organisation of the group. For example, questions remain about whether group control is best placed in the hands of an external conducting player (such as the role of a mix engineer in a live sound scenario), or whether issues of control should be managed directly by performers themselves. The answer is, as would be expected, that it depends on the needs of a given performance or group, but it can be said that further use of the system by composers would help to clarify some of these issues.

4. Conclusions

This project has chronicled the conception and development of Inclusive Interconnections, a parameter-sharing system which can be considered both inclusive and mutable, in that it supports a diverse range of approaches to performance whilst enabling interdependencies between members of the group to be flexibly reconfigured by players. This bottom-up approach stands in marked contrast to much existing work in the field, where the prevailing approach is to fix the model of interconnection at design level.

In the short term, it is hoped that this project highlights the value of moving towards an open system of interconnection for laptop performers, which allows players to bring their respective skills and experiences to interdependent performance. In the longer term, it is hoped to support wider diversity using the system, by addressing support for cross-platform operation, MIDI integration, and non-standard hardware configurations such as netbooks. From a social standpoint, inclusivity can be better supported by allowing players to leave and rejoin the network at any time, and by tailoring the richness of the system to suit both performer and composer-oriented use.

In order to extend mutability further, it is hoped to improve the overall intuitiveness of the model of interconnection in terms of its visual representation, whilst also allowing models to be manipulated through direct engagement with the graphical user interface. In particular, shared models of interconnection have highlighted a need to look again at the nature of reciprocal control, in order to identify methods which may prove more socially satisfying in a performance context.

Throughout this period, development of the system must continue to occur alongside sustained engagement by players. The challenge of designing interdependent systems can be seen as one which should facilitate interactions at group level, whilst at the same time drawing on the aims and approaches of the ensemble to further inform the design process. In turn, the challenge for individual performers is to develop their own practice in the context of an interdependent approach, both in terms of designing new instruments and in consideration of new ways in which the an ensemble may be organised socially. This can be seen as a process of narrowing the gap at either end, to provide a more general understanding of the creative challenges and possibilities of performing in an interconnected way.

5. Bibliography

Barbosa, A. (2003). Displaced Soundscapes: A Survey of Network Systems for Music and Sonic Art Creation. *Leonardo Music Journal*, *13*, 53-59.

Booth, G. (2010). Tracing Faultlines: Strategies for Developing Locally-Networked Ensemble Compositions. Retrieved December 30, 2010, from helios.hud.ac.uk/u0765940/faultlines/Faultlines.pdf

Brown, C. and Bischoff, J. (2002). Early Concerts of the League. *Indigenous to the Net*. Retrieved December 30, 2010, from http://crossfade.walkerart.org/brownbischoff/league_texts/early_league_concerts_f.html

Bukvic, I. (2010, October 22). What is L2Ork? *L2Ork Linux Laptop Orchestra*. Retrieved December 30, 2010, from http://l2ork.music.vt.edu/main/?page_id=5

Candy, L. (2005). Constraints and Creativity in the Digital Arts. Retrieved April 3, 2011, from http://creativity-embodiedmind.com/downloads/papers_workshop2/Linda_Candy_Paper.pdf

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.

Gurevich, M. (2006). JamSpace: Designing a Collaborative Networked Music Space for Novices. *Proceedings of the 2006 Conference on New Interfaces for Musical Expression*, 118-123.

Harker, A., Atmadjaja, A., Bagust, J. and Field, A. (2008). The worldscape laptop orchestra: Creating live, interactive digital music for an ensemble of fifty performers. *Proceedings of the 2008 International Computer Music Conference, Belfast, Northern Ireland.*

Hewitt, S. (2009). The Laptop As An Ensemble Instrument: Methods And Concerns. *University of Huddersfield Research Festival, 23rd March - 2nd April 2009, University of Huddersfield.*

Hewitt, S., Tremblay P. A., Freeman, S. and Booth, G. (2010). H.E.L.O.: The Laptop Ensemble as an Incubator for Individual Laptop Performance Practices. *Proceedings of the International Computer Music Conference, New York*, 304-307.

Kim-Boyle, D. (2008). Network Musics: Play, Engagement and the Democratization of Performance. *Proceedings of New Interfaces for Musical Expression Conference 2008, Genova, Italy.*

King, N. and Horrocks, C. (2010). *Interviews in qualitative research.* Los Angeles, CA: Sage.

Morgan, D. L. (1988). *Focus groups as qualitative research.* Newbury Park, CA: Sage.

Muller, R. (2006, June 16). OSCBonjour for Max/MSP. *Remy Muller - Ircam development blog*. Retrieved December 31, 2010, from http://recherche.ircam.fr/equipes/temps-reel/movement/muller/index.php?entry=entry060616-173626

Networked Music Review. (2007, May 2). *Turbulence.org*. Retrieved December 30, 2010, from http://www.turbulence.org/networked_music_review/

Neville, B. (2006, October 23). Networking: Max talking to Max. *Cycling 74 - Tools for Media*. Retrieved December 30, 2010, from http://cycling74.com/2006/10/23/networking-max-talking-to-max/

Pazel, D. et al. (2000). A Distributed Interactive Music Application using Harmonic Constraint. *Proceedings of the 2000 International Computer Music Conference, San Francisco*, 113-116.

Pestova, X., Donald, E., Hindman, H., Malloch, J., Marshall, M. T., Fernando, R., Sinclair, S. D., Stewart, A., Wanderley, M. M. and Ferguson, S. (2009). The CIRMMT/McGill Digital Orchestra Project. *Proceedings of the 2009 International Computer Music Conference, Montreal, Canada.*, 295-298.

Rovan, J., Wanderley, M., Dubnov, S. & Depalle, P. (1997). Instrumental Gestural Mapping Strategies as Expressivity Determinants in Computer Music Performance. *Proceedings of the Associazione di Informatica Musicale Italiana International Workshop 1997*, pp. 68-73.

Schnell, N., Borghesi, R., Schwarz, D., Bevilacqua, F and Mü ller, R. (2005). FTM - Complex Data Structures for Max. *Proceedings of the International Computer Music Conference, Barcelona, Spain, 2005.*

Smallwood, S., Trueman, D., Wang, G. and Cook, P. R. (2008). Composing for Laptop Orchestra. *Computer Music Journal*, *32*(1), 9-25.

Surges, G. and Burns, C. (2008). Networking infrastructure for collaborative laptop improvisation. *Proceedings of the 2008 Spark Festival, Minneapolis, Minnesota.*

Weinberg, G. (2002). Playpens, Fireflies and Squeezables - New Musical Instruments for Bridging the Thoughtful and the Joyful. *Leonardo Music Journal*, *12*, 43-51.

Weinberg, G. (2005). Interconnected Musical Networks - Towards a Theoretical Framework. *Computer Music Journal*, *29*(2), 23-39.

Winkler, T. (1998). Composing Interactive Music: Techniques and Ideas Using Max. Cambridge, MA: MIT Press

Wolek, N. (2010, April 13). The MPG Carepackage: coordinating collective improvisation in Max/MSP (Extended Abstract - Submission for SEAMUS 2010). *Lowkey Digital Studio*. Retrieved December 30, 2010, from www.nathanwolek.com/docs/2010/mpgcarepackage_abstract.pdf

Wyse, L. and Mitani, N. (2009). Bridges for Networked Musical Ensembles. *Proceedings of the International Computer Music Conference 2009, Montreal, Canad*

6. Appendices

A. Inclusive Interconnections: Software

i. Standalone Application

(See accompanying DVD-ROM for the application itself.)

Inclusive Interconnections v1.0 Application Notes

- requires an operational incoming network connection
- Inclusive Interconnections requires you to connect your own instrument
- see inclusive.interconnctions.mypatch.maxpat for an example instrument implemented in Max/MSP

Port Communications

Communications take place by default on the following ports:

- 9000 - IP selftest (UDP) selftest
- 9001 - network registration (multicast)
- 9002 - network registration and unicast communication (UDP)
- 9010 - local communication from your instrument input (your patch to the Inclusive Interconnections Application)
- 9011 - local communication to your instrument output (Inclusive Interconnctions Application to your patch)

See inclusive.interconnctions.mypatch for more details.

ii. Max/MSP Code

(See accompanying DVD-ROM for the code itself.)

Inclusive Interconnections v1.0 Code Notes

- Requires an operational incoming network connection
- Run inclusive.interconnections.multi to simulate operation of multiple clients

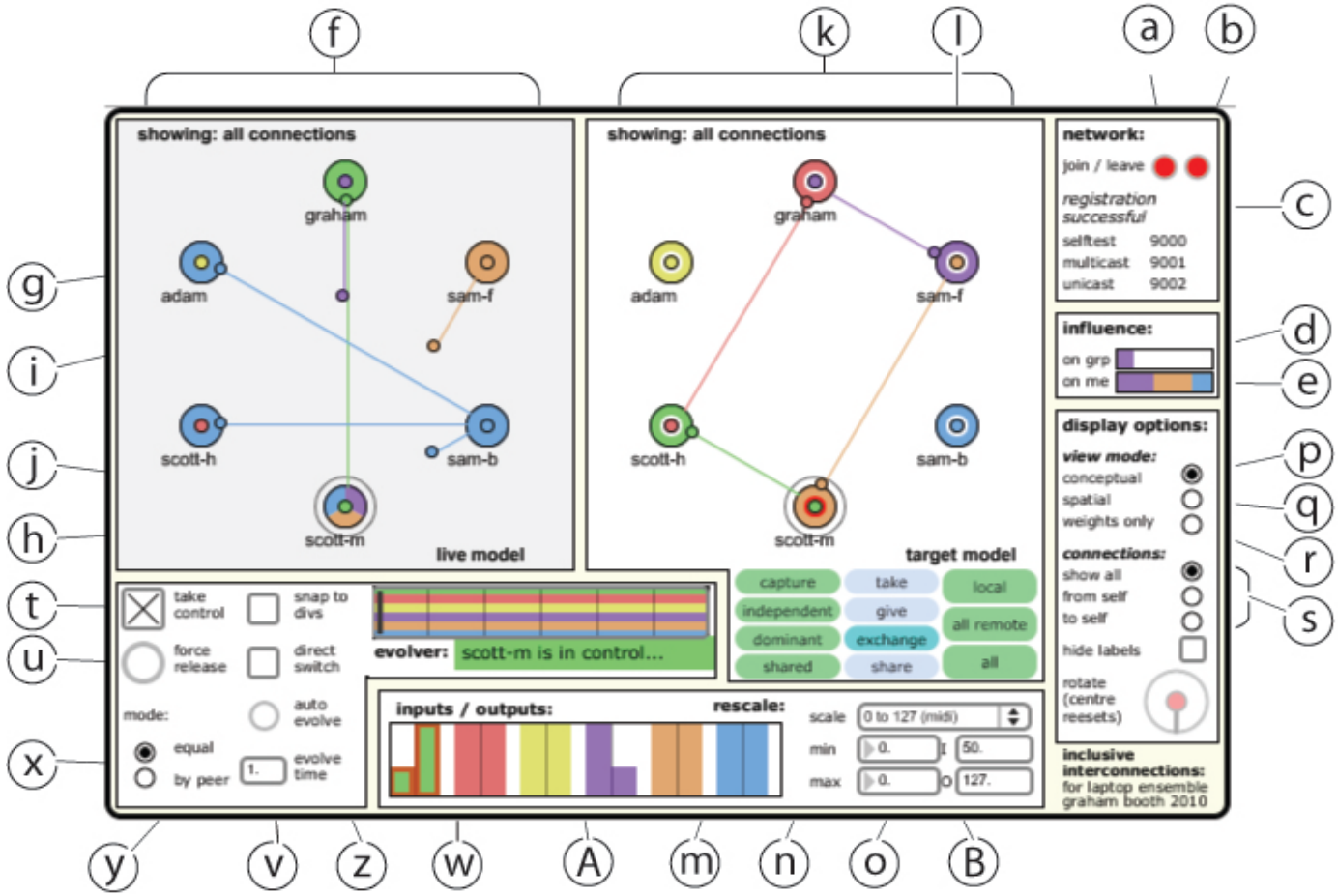
Dependencies

- Requires FTM.2.5.0.BETA.15-Max5 (later versions may cause problems)
- Requires OSC-route.mxo

For more information see section iv. Glossary of Max/MSP Abstractions

iii. Documentation for Users

Overview of the Inclusive Interconnections Graphical User Interface:



Key to Graphical User Interface Elements:

NETWORK:

(a) **join** - click to join the network (red = disabled)

(b) **leave** - click to leave the network (only if you have no influence over others - see "on grp" influence bar)

(c) **status** - shows network registration status

INFLUENCE:

(d) **on grp** - shows your influence on the group (colours = who, size = how much)

(e) **on me** - show the influence of the group on you (colours = who, size = how much)

LIVE MODEL:

(f) **live model display** - shows the currently applied mode of interconnection (this is the same for all players and cannot be manipulated)

(g) **inner core** - shows identifying colour of each player

(h) **outer ring** - shows who holds influence over a player

(i) **connecting lines** - show who a player is controlling and by how much

(j) **outer circle** - identifies your player (always at the bottom by default)

TARGET MODEL:

(k) **target model display** - allows you to create an ideal model of interconnection. this is different for all players.

(l) **inner core** - click on to select player (or drag a box around multiple players). black ring shows currently selected players

(m) **preset target models** - set up a preset model to work from, includeing ability to capture current live model (must not be in control)

(n) **select tranformation type** - take, give, exchange or share influence with other players to create a new target model (must not be in control)

(o) **apply transformation to...** - determines whose influence will be involved in the transformation

DISPLAY OPTIONS:

(p) **conceptual view mode** - shows players arranged equally around the centre of the display

(q) **spatial view mode** - allows players to be moved to create your own view (see also rotate dial)

(r) **weights only view mode** - displays a bar graph showing the influence of each player

(s) **connections -** filter connections between players to show only ones connecting to or from you

EVOLVER:

(t) **take control** - take control of the live model

(u) **force release -** force another user to release control of the live model

(v) **direct switch** - switch instantaneously from the target model to the live model (must be in control first)

(w) **slider** - scrub between the target model and the live model (must be in control first)

(x) **mode** - change the traversal mode ("equal" interpolates in parallel, "by peer" interpolates one by one)

(y) **auto evolve** - fade smoothly over a set time between the target model and the live model

(z) **snap to divs -** snap to divisions based on the number of players (stepped output - affects slider and auto-evolve controls only)

INPUT / OUTPUT

(A) **in out bars** - show each player's current input and output value (own highlighed by a red box)

(B) **rescale -** set the input and output range (0. to 1., 0. to 127. or set own range)

iv. Glossary of Max/MSP Abstractions

1. Master Patches	
inclusive.interconnections.mypatch	Example instrument patch which interfaces with the inclusive.interconnections.single locally via UDP
inclusive.interconnections.multi	Top level patcher which runs multiple copies of inclusive.interconnections.single patch, simulating a distributed network containing eight peers
inclusive.interconnections.single	Master client patch from which the standalone application is built
inclusive.interconnections.table.test	Patch for testing network registration (simulates a distributed network of four peers)
2. Matrix Calculations (*.mut)	
mut.calc.outputs	Calculates a matrix of output values for each player based on current input values and live model weights
mut.clickdrag	Translates x,y and click data from the target model display into selection and movement operations
mut.evolver	Controls interpolation between the current live model and the chosen target model
mut.gen.archetypes	Generates archetypal target models, which provide useful starting points for further manipulation
mut.rescale	Rescales input and output values from/to a users instrument patch, according to a range set on the user interface.
mut.state.get	Gets the current state of the inputs, weights, spatial-positions and incontrol status from existing players (activated on registration)
mut.state.set	Sets the current state of the inputs, weights, spatial-positions and incontrol status across the network (all players)
mut.takecontrol	Interprets the currently in control player as either "local", "remote" or "no-one" causing correct user interface panels to be enabled/disaled And current controlling player status to be displayed
mut.tomatrix	Uses the output of net.table (relative index) to add or delete rows and/or columns from local matrices when a new peer joins or leaves the network.
3. Network Infrastructure (*.net)	
net.all	Wrapper for all net. Abstractions (handles peer to peer registration)
net.local.id.get	Requests username, to be used as local identifer for all table lookup operations and OSC communications, making communications human readable and easily attributable to a specific user. If no name is entered, a random untitled one is given.
net.local.ip.get	Detects and outputs IP addresses of local network adapters, ignoring local loopback address (abstraction based on mxj net.local)
net.local.ip.test	Tests available network adapters one by one by

	establishing a unicast UDP channel for each. The first adapter to successfully receive an incoming message is deemed as the best connection to use.
net.local.ip.test.unicast.multi	Sets up communciations channels for testing detected network adapters (allows multiple udpsend objects to be used in parallel using poly~, thus avoiding problems with fast-switching)
net.table	Handles registration and de-registration of peers on the network, maintaining an equivalent list of registered peers on each client.
net.transport.multicast	Used in the registration process to enable multicast communications between players (wrapper for mxj net.multi.send / net.multi.recv)
net.transport.peer	handles communication between registered peers. OSC messages are prepended by the absolute ID of the player to send to (with -1 referring to all players).
net.transport.unicast.multi	sets up communciations channels for the peer transport, allowing multiple UDPsend objects to be used in parallel (using poly~), thus avoiding problems with fast-switching.
net.transport.unicast.single	abstraction handling all single channel UDP network traffic (gate suppresses outgoing messages when not required, port is settable via argument, option to print to max window to debug)
4. User Interface (*.vis)	
vis.evolver.bar	displays parallel or serial colour bars underneath the evolver slider, depending on evolver mode chosen (equal, or by peer)
vis.influence.group.on.self	handles drawing of "on me" influence bar
vis.influence.self.on.group	handles drawing of "on group" influence bar
vis.inout.bars	handles drawing of input/output bar pairs for each player, based on the values in the input and output matrices
vis.model	handles drawing of all elements of the live and target model displays
vis.positions	calculates conceptual and spatial positions, as well as rotated versions of these for output
vis.user.interface	bpatcher which combines all elements of the user interface

B. Inclusive Interconnections: Demonstration Videos

(See accompanying DVD-ROM for the videos themselves.)

i. Inclusive Interconnections Lab Demonstration

The demonstration video comprises the following sections:

a) Networking Lab Test (0:00 - 17:45)
b) User Interface Demonstration (17:45 - 39:08)
(this video is organised into chapters, which correspond to the headings in section 7)

ii. Real World Test Performance with H.E.L.O.pg

See "Real-world Test Performance with HELOpg - Full Group.avi" for the full test performance, captured on DV camera. Also provided are the following simultaneous screencasts of the same performance. For best results, play all four videos side-by-side, starting one after the other as quickly as possible.

- "Real-world Test Performance with HELOpg - Adam Jansch.m4v"
- "Real-world Test Performance with HELOpg - Sam Freeman.m4v"
- "Real-world Test Performance with HELOpg - Scott Hewitt.m4v"
- "Real-world Test Performance with HELOpg - Scott McLaughlin.m4v"

C. Audio Recording of Evaluation Session

(See accompanying DVD-ROM for this digital audio recording.)

The recording itself documents a feedback discussion, chaired by the author, which took place with members of H.E.L.O.pg. The format of the discussion is detailed further in section 7.4. Total duration is 1 hour 14 minutes, and the members participating in the discussion were Sam Birkhead, Sam Freeman, Scott Hewitt and Adam Jansch.