bf-pd: Enabling Mediated Communication and Cooperation in Improvised Digital Orchestras

Luke Dahl[1], Florent Berthaut[2], Antoine Nau[2], and Patricia Plenacoste[2]

[1] University of Virginia
[2] Université de Lille
lukedahl@virginia.edu florent.berthaut@univ-lille1.fr
antoine.nau@etu.univ-lille3.fr patricia.plenacoste@univ-lille1.fr

Abstract. Digital musical instruments enable new musical collaboration possibilities, extending those of acoustic ensembles. However, the use of these new possibilities remains constrained due to a lack of a common terminology and technical framework for implementing them. Bf-pd is a new software library built in the PureData (Pd) language which enables communication and cooperation between digital instruments. Its design is based on the BOEUF conceptual framework which consists of a classification of modes of collaboration used in collective music performance, and a set of components which affords them. Bf-pd can be integrated into any digital instrument built in Pd, and provides a "collaboration window" from which musicians can easily view each others' activity and share control of instrument parameters and other musical data. We evaluate the implementation and design of bf-pd through workshops and a preliminary study and discuss its impact on collaboration within improvised ensembles of digital instruments.

Keywords: Digital Orchestras, Laptop Orchestra, Digital Musical Instruments, Collaboration, BOEUF, bf-pd, PureData

1 Introduction

Musical instruments based on electronic and digital technologies enable interactions between musicians that acoustic instruments do not typically afford. Control of a single instrument can be shared between multiple musicians, or a musical output from one instrument can be used as a control or input to another instrument. Ensembles such as the League of Automatic Composers and The Hub have been exploring the collaborative potential of digital technologies since at least the 1970's [5]. Contemporary ensembles, such as laptop orchestras, continue to use and extend these collaborative possibilities. However, it seems to be the case that composers and musicians who do utilize new modes of collaboration often build their own bespoke system for each composition or ensemble, thus re-implementing common functions and capabilities. We believe this is due to two factors. First, there is not a common language or terminology for labelling and discussing the ways in which musicians collaborate. Second,

there is not a common widespread technical infrastructure for implementing these various modes of collaboration. This situation is especially problematic for spontaneously formed ensembles of heterogeneous instruments, where the lack of a quickly integrated system for sharing data between instruments leads to these interactions being neglected. New technologies may also impede interactions that were simple or easily available in acoustic ensembles. For example, a musician in a digital orchestra may find it difficult to discern which of their fellow musicians is generating a specific sound. Thus, systems for musical collaboration may need to enable digitally mediated communication for activities that were previously unmediated.

1.1 Contribution

In [1] we presented BOEUF, a conceptual framework and set of components for describing digital orchestras and classifying the *modes of collaboration* they make possible. We developed this classification after conducting a survey of digital ensembles, collaborative instruments, and other frameworks and surveys such as [3][6][11].

In this paper we present bf-pd, an implementation of the BOEUF components in the PureData (Pd) language. Musicians working in Pd can integrate components of bf-pd into their instruments, and thus gain access to a subset of the BOEUF collaboration modes, operated through a generic graphic interface. Specifically, bf-pd is designed to facilitate cooperation and sharing of control data between musicians, and to increase awareness by making musicians' activity visible to each other. Bf-pd is the first implementation of the BOEUF components, and the first system to explicitly support the BOEUF modes of collaboration. It allows us to test the usability of the components and the design of the protocol before going on to implement these in other software or hardware systems. It also allows us to evaluate how the use of these components affects musical collaborations. Bf-pd relies on a protocol of Open Sound Control (OSC) messages to communicate between instruments. This protocol can be used by future implementations of the BOEUF components on other platforms.

First we review the BOEUF conceptual framework. Then we describe the components currently supported by bf-pd, their interfaces and how to integrate them into a Pd instrument, and the details of the communication protocol. We also present the 'collaboration window', a graphical interface for controlling and displaying communication and cooperation between instruments. Lastly, we discuss preliminary results from a series of workshops and a pilot study we conducted.

1.2 Related Work

Monad [4] is an example of a recent networked musical collaboration, with game-like interaction and scoring, and a graphical UI for displaying user activity. Several protocols and software tools have been created to deal with the sharing of musical data for both single instruments and within networked orchestras. For

example, Jamoma [10] and libMapper [8] both give access to the structure and parameters of networked instruments, sometimes with features for watching and grabbing parameters. An interesting example is the Digital Orchestra Toolbox [7] which simplifies the collaborative creation and mapping of digital musical instruments (DMI). Most of these tools in turn rely on the OSC protocol for network communication. However, while they provide all the generic sharing and mapping features required for networked musical control, these tools do not specifically cover all the modes of collaboration used in digital orchestras, and thus fail to provide a common basis for creating orchestras of mixed DMIs. Unlike these other software frameworks, bf-pd is based on a conceptual framework, which is intended to cover the range of known collaboration possibilities. We used this framework to guide the design of bf-pd, and to evaluate its success.

2 The Boeuf Framework

BOEUF is a conceptual framework for modelling and building orchestras of DMIs. It consists of a classification of *modes of collaboration*, as well as a set of *components* which can be used to enable these modes in DMIs. ("Boeuf" is french slang for a jam session.) Here we briefly summarize the framework, which is presented in more detail in [1], with a focus on the bf-pd implementation.

2.1 Modes of Collaboration

After conducting a survey of ways in which musicians in both acoustic and digital ensembles work together during a musical performance, we formulated three categories of modes of collaboration to describe these activities.

Cooperation modes describe the coordination of musicians' actions with respect to their instruments. *Independent* cooperation occurs when each musician controls their own instrument while playing together. *Complementary* cooperation occurs when two or more musicians can affect different aspects of the same musical output. For example, in a digital orchestra one musician might change the pitch of an instrument while another controls the timbre. *Concurrent* cooperation occurs when multiple musicians can affect the same musical output at the same level, i.e. when they modify the same parameter on a single instrument.

Communication modes are ways in which musicians exchange information which may then influence their actions. This communication may or may not directly impact the production of sound. *Awareness* includes all non-intentional communication, such as the means by which musicians keep track of each others' activities. In acoustic ensembles awareness is usually non-mediated and is facilitated by musicians' ability to see each other's movements and to distinguish each other's sounds. However in digital ensembles, one's instrument may be unfamiliar to the other musicians and sound may not originate from the performer's location, so awareness must often be digitally mediated.

Indications are intentional communicative acts. These include commands, such as a conductor cuing an entrance, and suggestions, such as a nod or glance

near the end of a solo. In digital ensembles indication can be mediated in the form of symbolic messages between musicians. *Exchange* refers to transfers of musical data between musicians. These too can be mediated, such as sending MIDI or OSC data between devices, or non-mediated, as when an improviser riffs on a motif played by another musician.

Organisation modes do not have any effect on the music produced, but rather impact the communication and cooperation modes. *Nomination* consists of defining the roles of musicians within the orchestra, e.g. who is the leader. *Grouping* defines a hierarchy of groups of instruments. *Selection* is the act of choosing a single instrument or a group in the context of cooperation or communication, e.g. selecting which musician to send an indication to. In the work described in this paper we focus on enabling the Cooperation and Communication Modes of Collaboration. The bf-pd library does not currently address organisation explicitly.

2.2 Components

The BOEUF components comprise a generic model of a digital orchestra, and are designed to enable the BOEUF modes of collaboration described above.

A **session** represents an instance of a collaborative music-making ensemble or event. A piece for laptop orchestra, or a spontaneous jam session of digital musicians would take place within a session. A session contains instruments, and the network of possible interactions between them.

An **instrument** represents a bounded set of music-generating processes and a user interface (UI). An instrument may contain modules, parameters, and outputs, and it can send and receive messages. We presume that each musician in the orchestra is in control of at least one instrument. Thus, an instrument often acts as a proxy for the musician.

A **parameter** is an attribute of a module or instrument that influences its musical production, and which can be controlled through the instrument's UI. Parameters can be of various types, and in bf-pd these types are: *cont* (a floating point number between 0.0 and 1.0), *midi* (an integer between 0 and 127), *bang* (which can trigger an event), and *bool* (on or off). In bf-pd a parameter can also be a multiple of these types, e.g. a '4 cont' is 4-dimensional parameter, e.g. composed of four floating point values. A parameter can be:

- *Set* to a new value, either through the instrument's UI, or by other instruments (if the owner has granted access to do so).
- *Watched*, where the parameter value is sent to other instruments every time it is changed. (This functions as a means of both *awareness* and *exchange*.) In bf-pd all parameters are watched by all other instruments, and appear in the collaboration window (see 3.2).
- *Indicated*, where another instrument may propose but not set a new value for the parameter. In bf-pd this is done through the 'ask' functionality in the collaboration window.

– *Retrieved*, where the current value is returned once. This is not currently implemented in bf-pd.
– *Grabbed*, where a parameter can be set only by the instrument which has grabbed it. This is a strategy to deal with concurrent access to parameters, but is not currently implemented in bf-pd.

An **output** is a musical attribute that is produced by a module or instrument. They can have the same types as parameters. Outputs can be retrieved and watched by other instruments, and function as a means for both *awareness* and *exchange*.

A **meter** is a component of an instrument that is not used in the actual sound production, but rather indicates the activity of the instrument. Each instrument has an overall activity meter which is visible to other instruments through the collaboration window. Bf-pd does not currently have a separate Meter type.

A **module** is a component that produces musical data (either audio or control data), and can contain parameters, outputs, and meters. A module can have a type, which defines a set of parameters and outputs. For example all modules of type 'LowPassFilter' would have parameters for cutoff frequency and resonance. This would allow the state of entire modules to be exchanged. Modules are not currently implemented in bf-pd.

A **group** is a set of instruments or other groups, and is used for organisation modes. For example, the parameters common to all instruments in a group could be set simultaneously, or a message could be sent to all instruments in a group. Groups are not currently implemented in bf-pd.

A **message** can be a text, image, or video sent from one instrument to another instrument or group. Messages are not currently implemented in bf-pd.

3 bf-pd

3.1 Integrating the components

Bf-pd allows musicians to access modes of collaboration available in the BOEUF framework with their instrument developed in PureData by integrating abstractions that implement the components of the framework.

The first abstraction to add is *bf-instrument*, with the name of the instrument as an argument. As seen in Figure 1.a, this object displays an activity meter and manages the collaboration window (see Section 3.2). The show_others toggle allows for displaying or hiding other instruments in the collaboration window.

Then for each parameter they want to share with other musicians, users create a *bf-param*. Arguments of this object are the instrument name, the parameter name, the number of values it has, and its type. Once created, a GUI is generated for the parameter, with widgets to control it directly with inlets and outlets for setting the values and retrieving them, as shown in Figure 1 (b and c). For any musical output that they want to share, they add a bf-output. This object has the same argument and possible types as bf-parameter, but only has inputs.

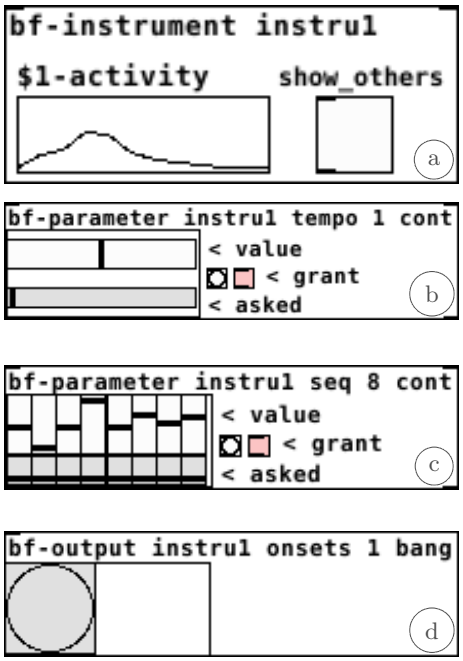

Fig. 1: Some bf-pd components: a) bf-instrument and activity meter, b) bf-parameter of type *cont*, c) bf-parameters of type *8 cont*, d) bf-output of type *bang*

Finally, musicians create a bf-session object. This object handles connecting with other instruments on the network. It has two arguments: the name of the instrument (or * if several instruments are opened), and the name of the session (which is used to filter messages so that multiple sessions can take place on the same network).

Figure 2 shows an example patch of an instrument with two parameters, one with a single continuous value and one with eight continuous values, and one output with a bang type which is used to share onsets to other musicians.

3.2 Collaboration window

Bf-pd creates a *collaboration window* for each instrument (see Figure 3), which functions as the primary UI for digitally mediated modes of cooperation and communication. The leftmost column displays the activity meter and all bf-parameters and bf-outputs of the instrument. The columns to the right display the activity meter, bf-parameters, and bf-outputs of each other instrument in the bf-session.

The activity meters display each instrument's audio output as the energy in 12 bark-spaced bands. The meters are intended to improve the ability to tell

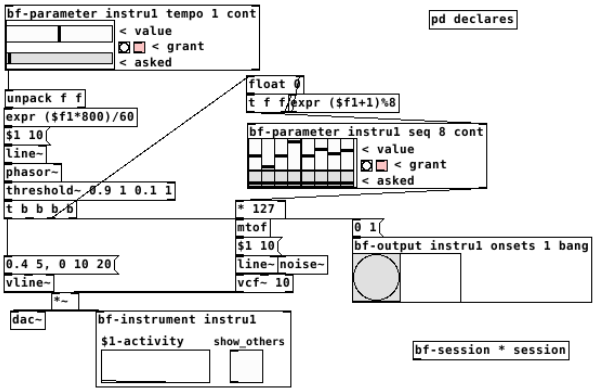

Fig. 2: The Pd patch for an instrument using bf-pd, with the components from Figure 1

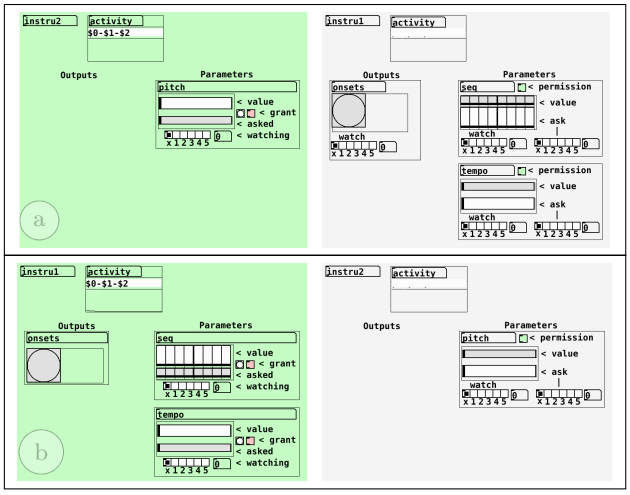

Fig. 3: Collaboration windows of bf-pd: a) for instrument "instru1", b) for instrument "instru2" . In the left columns are the activity, parameters and outputs of one's own instrument, in the right columns are those of the other instrument in the session.

who is making which sound, and thus increase *awareness*. The collaboration window UI for each parameter affords a number of communication and cooperation modes. For parameters belonging to other instruments, the user can see a parameter's value as its owner changes it (*awareness*). The user may also 'ask' for a new value (*indication*), and the parameter's owner will see the asked value in their collaboration window. If the owner grants permission, asking for a new value will change the parameter (thus enabling both *concurrent* and *complementary* coop-

eration). *Exchange* can be enabled in two ways. The user can 'watch' another's parameter by selecting a *watch-bus* with the radial button on the parameter's UI. If the user selects that same bus with the 'watching' selector on one of their own parameters, their parameter will now be controlled by the other's parameter. Like parameters, the outputs of other instruments can also be watched. To send information from one's own instrument to another's parameter, one can send data to an *ask-bus* through a bf-ask object in one's own instrument. Then select this same bus in the 'ask' selector of the other instrument's parameter. The UI for all of the actions described here can be seen in Figure 3.

3.3 Implementation

All communication between instruments is done using OSC messages. This ensures that the implementation of BOEUF is not limited to PureData, but can be later extended to other software instruments through plug-ins that will parse these messages. The main messages are listed in Figure 4. Collaboration messages in particular are composed of the session name, the instrument name, the component type (parameter, activity, or output) and name and the type of action (set, ask, or watch). For ask and set messages, the arguments of the message are the index of the value, the value asked or set, and the emitter of the message. The graphical user interface for both the abstractions added to the instrument patch and the collaboration window are generated using the dynamic patching and graph-on-parent capabilities of PureData.

/boeuf/session/request	discover other instruments
/boeuf/session/hello ip_address	inform others of our ip address
/boeuf/session/instru/parameters/param/declare nb-values type	provide information on parameter
/boeuf/session/instru/parameters/param/set index value set-by-name	set the value for a parameter
/boeuf/session/instru/outputs/out/set index value set-by-name	set the value for an output
/boeuf/session/instru/parameters/param/ask index value asked-by-name	ask a value for a parameter

Fig. 4: Main Open Sound Control messages for bf-pd

4 Evaluation

In this section, we present some preliminary results from a series of evaluations of bf-pd and its impact on improvised digital orchestras.

4.1 Workshops and iterations

We conducted three workshops with both non-musicians and musicians. Each began with a PureData tutorial to ensure that everyone was familiar with the interface and concepts. We then asked musicians to play in groups of three for five minute improvisation sessions using an instrument that we provided. Each group played a session for each of three conditions: a) without bf-pd, b) with only the communication part of bf-pd (i.e. watching others' activity in the collaboration window without interacting with them) and c) with both the communication and cooperation possibilities of bf-pd (i.e. interacting using the collaboration window). We use this same protocol in the study presented below. We filmed each session and subsequent discussions to get feedback on each condition.

We used this feedback to iterate on the implementation of bf-pd and on the design of the collaboration window. At the time of the first session, musicians could control their own parameters only in the instrument patch, and could view and access others' parameters in the collaboration window. They commented on the difficulty caused by frequently switching between the two windows. As a result, we added to the collaboration window a dedicated column to display and control one's own parameters and outputs. Now musicians can both play their instrument and collaborate without leaving the collaboration window.

At the time of the second workshop, the routing of watched parameters and outputs had to be done by modifying one's instrument patch, and was not possible from the collaboration window. Participants told us they wanted to experiment with different routings while playing. We then added the ability to connect watched parameters and outputs to one's own parameters without leaving the collaboration window (through the *watch-bus*). We also expanded the OSC protocol so that messages provide more information on cooperation, e.g. who is asking a parameter, who is watching someone's parameter, and so on. This allows us to analyse more precisely the impact of various bf-pd design choices.

4.2 Preliminary study

After integrating the feedback from the workshops we conducted a preliminary study to investigate the effects of using bf-pd on musicians' interactions. Our study participants were 7 electronic musicians with an average 8.6 years of experience (sd=5.5).

Our study was structured the same way as the workshop sessions: a Pd tutorial, followed by three sessions of improvisation of at least 5 minutes with an instrument we provided. The interface for this instrument was composed of one output and five parameters, including a repeating pattern of eight values driven by a tempo parameter. Musicians were randomly grouped by 3 during the various sessions. Unfortunately, due to technical issues we were only able to measure one group in all three conditions. We compared the three same conditions as in the workshops: In Condition *NO*, the musicians played only their own instrument. In Condition *COM*, they could see the other instruments in the group, but were instructed to control only their own instrument. In Condition *COOP* they were

encouraged to cooperate by controlling others' parameters and granting access to their own parameters. All interactions took place in the collaboration window. This allowed us to analyse the separate impact of *communication* (visualising others' activity) and *cooperation* (actually interacting with others' instruments).

4.3 Analysis

During the study we recorded each time a musician changed one of their own parameters. We recorded video of all sessions. And we asked participants to fill out a questionnaire after each condition. A five-level Likert scale was used to evaluate different aspects of the participants' experience.

Parameter Analysis From the recorded data we find that participants changed the values of their own parameters more frequently in *COM* (1.17 times per second on average) and *COOP* (1.0) compared to *NO* (0.61). This suggests that participants were more engaged with the interface when communication was enabled. The fact that changing one's own parameters was less frequent with cooperation may suggest that some attention was on other players' parameters instead of their own. We also calculated to what degree participants manipulated one parameter versus manipulating all the parameters available to them. The results show that as participants moved from *NO* to *COM* to *COOP* they focused on fewer and fewer of their own parameters. This is interesting because it may suggest that participants were more engaged and focused on how their actions affected and interacted with the actions of the other musicians (as opposed to haphazardly changing parameters). These two results could also be due to the increasing complexity of the interaction and higher cognitive load when moving from one condition to the next.

Video Analysis From the recorded video we notice that participants seem to look at each other less often in Conditions *COM* (13) and *COOP* (11), compared to *NO* (19). We also saw that digitally mediated actions could effect non-mediated interactions. For example, occasionally participants would spontaneously laugh or move at the same time in response to something happening in their instruments. And we noticed that in *COOP* there was more verbal communication between participants, as they discussed strategies for synchronizing their tempo or provided explanations to each other.

Questionnaire Analysis Our questionnaire results come from two groups: one from the third workshop and the other from the preliminary study. We found that musicians in *COM* (score=4) could better distinguish between the activities of the other musicians than they could in *NO* (3.8). The question was not asked for *COOP*. We asked to what degree participants felt like they were making music together with the other musicians. Musicians felt equally together in Conditions *NO* and *COOP* (4.3), but less together in *COM* (3.5). We asked whether each

condition was a better shared experience than the former, and found that *COM* was not better than Condition *NO* (2.6 out of 5), while *COOP* was better than *COM* (4.1/5).

5 Discussion

We note that our workshop and study results are "preliminary" in the sense that we do not have enough participants to generate statistically significant results, and the order of conditions was not randomized. However, they did generate valuable feedback which we used to improve bf-pd and which suggests directions for further investigation.

It is difficult to interpret some of the results. Did participants look at each other less in Conditions *COM* and *COOP* because they could view others' activity and cooperate with them through the UI? Or was it because each subsequent condition demanded more attention? Or perhaps this is due to the way collaboration information is displayed. An improved study design may help answer these questions. So might further research into how to better display collaboration information.

Another limitation of our study is that the order of conditions is not randomized between groups. As the participants pass through each condition they may become more comfortable with their instrument and with the other musicians. This may have an effect on the measured variables. Having multiple groups with a counterbalanced order of the conditions would remove this effect.

The study does suggest that bf-pd is succeeding at enabling and encouraging communication and cooperation. Participants can more easily distinguish others' activities when using bf-pd, and so *awareness* is increased. A recurring comment was that participants felt like they were "sharing their instrument with each other", or even that they were "all playing the same instrument". This may be due in part to the common interface, where all instruments appear and can be controlled at once. It is also a demonstration of what can happen when musicians use bf-pd to move from *independent cooperation* to *complementary* and *concurrent cooperation*.

There are a few components from the BOEUF framework which are missing in bf-pd. Adding these might further enrich the experience of cooperative music-making in bf-pd. Messages could facilitate textual communication through the collaboration window. And, the ability to define groups and roles would facilitate organization within the orchestra. For example, one could take the role of "tempo master", so that others will watch their tempo parameter.

6 Conclusion

Bf-pd is a software framework that makes it easy for digital musicians working in PureData to access and change parameters on each others' instruments, to share data between instruments, and to perceive the actions of other musicians in the ensemble. One of the most challenging aspects of designing bf-pd was to create

user interfaces in Pd that quickly convey awareness information, and which make it easy to access the various modes of cooperation. Future work will focus on refining the UI, especially by investigating visualisations of musicians' activity, whether inside Pd or outside with network connected applications. Interaction design becomes even more challenging when we consider integrating bf-pd into embedded hardware platforms such as Bela [9] which lack GUIs. One solution might be to use augmented reality interfaces and allow musicians to modify the collaboration window and overlap it with their own unique physical interface [2], thus merging mediated and non-mediated communication. We currently have limited information on musicians' experience integrating bf-pd into their own music-making process, and so further user studies are needed.

A video of bf-pd can be seen at `https://vimeo.com/214380530`. Bf-pd will be released at `http://boeuf.hitmuri.net`. We invite you to use it and look forward to your feedback.

References

1. F. Berthaut and L. Dahl. BOEUF: A Unified Framework for Modeling and Designing Digital Orchestras. *Lecture notes in computer science*, 9617:153 – 166, Sept. 2016.
2. F. Berthaut and A. Jones. Controllar: Appropriation of visual feedback on control surfaces. In *Proceedings of the 2016 ACM on Interactive Surfaces and Spaces*, ISS '16, pages 271–277, New York, NY, USA, 2016. ACM.
3. T. Blaine and S. Fels. Contexts of collaborative musical experiences. In *Proceedings of NIME 03*, pages 129–134, Singapore, Singapore, 2003.
4. C. Cakmak, A. Camci, and A. Forbes. Networked virtual environments as collaborative music spaces. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, volume 16 of *2220-4806*, pages 106–111, Brisbane, Australia, 2016. Queensland Conservatorium Griffith University.
5. S. Gresham-Lancaster. The aesthetics and history of the hub: The effects of changing technology on network computer music. *Leonardo Music Journal*, pages 39–44, 1998.
6. I. Hattwick and M. M. Wanderley. A dimension space for evaluating collaborative musical performance systems, 2012.
7. J. Malloch, S. Sinclair, and M. Wanderley. A network-based framework for collaborative development and performance of digital musical instruments. In *Computer Music Modeling and Retrieval. Sense of Sounds*, volume 4969 of *Lecture Notes in Computer Science*, pages 401–425. 2008.
8. J. Malloch, S. Sinclair, and M. M. Wanderley. Libmapper:(a library for connecting things). In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 3087–3090. ACM, 2013.
9. G. Moro, A. Bin, R. H. Jack, C. Heinrichs, A. P. McPherson, et al. Making high-performance embedded instruments with bela and pure data. 2016.
10. T. Place and T. Lossius. Jamoma: A modular standard for structuring patches in max. In *Proceedings of the International Computer Music Conference*, pages 143–146, 2006.
11. G. Weinberg. Interconnected musical networks: Toward a theoretical framework. *Comput. Music J.*, 29(2):23–39, June 2005.