AUTOMATED AD HOC NETWORKING FOR MOBILE AND HYBRID MUSIC PERFORMANCE

Georg Essl

University of Michigan
Electrical Engineering & Computer Science and Music, Ann Arbor, USA
gessl@eecs.umich.edu

ABSTRACT

Zero-configuration networking is a helpful device to mitigate numerous problems of manual configuration of connections of network connection in performance. We discuss how this technique can be used to create a networked name-space. This helps organize networked music performance making them robust and easy to set up and use. Participatory networking which scales becomes easy to achieve and explicit addressing is made unnecessary.

1. INTRODUCTION

The purpose of this paper is to describe how network connectivity can be organized in a seamless and automated fashion between diverse music oriented environments and hence ease networking and interoperability. Network service discovery deals with the problem of bringing entities in a network together to utilize some networked functionality. We will describe how existing network service discovery solutions can be reinterpreted to serve as seamless and automated mechanism to organize networked music performances, especially in a local network.

The availablility of local and global network access is ever increasing. The cost of a wireless router is low and hence it is very simple to create a local wireless network virtually anywhere and commodity devices such as laptops or mobile smart phones have WIFI capability built in. Hence it also becomes much easier to conceive of computer music pieces and performances that use networks, specifically local area networks. In order for two devices to connect on a network, they have to know each others network addresses, such as an IP number. However this task of connecting is not robust. If one transitions from one wireless network to another the IP numbers in use may change. One common strategy in response to this is to use static and device-centric IP numbers. However this in turn makes the setup fragile with respect to device failure. If a failed device is to be replaced it has to occupy the same IP number and requires the needed configuration.

Mechanisms have been invented that allow devices to find each other under conditions of changing networks and changing devices. The basic concept is called Zero Configuration Networking [7] and its goal is to allow discovery rather than configuration of desired network connectivity.

Networking has been the focus of interest in computer music performance for a long time and numerous key issues have been addressed. Primary focus of much of the research has to do with long-range networking and the associated network latency and synchronization issues [3, 4]. However connectivity discovery, and specifically its use to organize complex networked setups, has received little attention.

Our own motivation for this work springs from enabling local wireless networking for mobile music performances. Networking of mobile devices has also emerged in recent years. In the commercial realm it is used to enable server-centric communication. For example Smule-applications communicate music performance and other information between networked participant through a server architecture [12]. The Stanford Mobile Phone ensemble has used networking as part of performances [9] and networking support can be found in both urMus [5] and MuMo [2].

2. NETWORK SERVICE DISCOVERY

The idea behind zero-configuration network service discovery is to broadcast the availability of a service over the network [7]. The key information to transmit is the network address to connect to for a desired service. Hence rather than trying to connect to addresses one connects to available services.

Numerous specific implementations have been suggested and for practical consideration we will focus on Bonjour, which is an OpenSource solution advanced by Apple [1]. Bonjour is already integrated in the development framework for iOS devices and can in principle be ported to other devices.

The basic mechanism for service discovery has two roles. One is that of offering a service. For this a network service is created that is responsible to advertise the service to the network. The advertisement of a service consists of a triplet of information. The network service identifier (NSID), a human-readable descriptive name of the service, and a network port. It is implied that the device advertising the service is the one allowing connection to it. Broadcasting repeatedly informs the network of the currently available services. If network service is active, it advertises its triplet to the network. These can then be

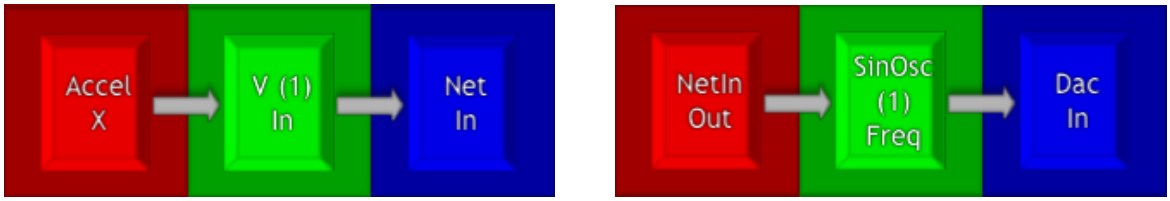

Figure 1. Complementary urMus dataflow patches that use NetOut and NetIn. In this example the x-axis accelerometer is controlling a symmetric gesture on one device, and the control is sent to the other to modify the frequency of a sine oscillator which is then played back.

found and collected by other devices on the network. If the advertised service is of interest, a connection can be initiated with the information provided by the IP address of the broadcasting device.

A key component for the matching is the NSID. In Bonjour this ID has the following format `_ServiceID._NetworkProtocol`. Where `ServiceID` and `NetworkProtocol` are placeholders. `NetworkProtocol` defines the network protocol in use and can currently be `tcp` and `udp`. In our case we only use udp connections. `ServiceID` is a flexible string with some restrictions that is a unique identifier of the service provided. For example `_Net1urMus._udp.` is a valid NSID.

2.1. Name-spaced Network for Music Performance

The ServiceID string is arbitrary and hence can be used to organize services. For the purpose of this work, we reinterpret the ServiceID as a name-space identifier.

There are multiple motivations for this proposal. If the name space is properly defined, one can construct networked performances that utilize multiple network discovery mechanisms to organize the network structure of the piece itself. Furthermore, one can then also envision multiple concurrent performances of the same or different performances with non-interfering configurations.

Our implementation uses the convention that the ServiceID is a concatenation of namespace components. The rightmost part of the ID is an applicationID. Left of the applicationID there is an optional instance counter. The instance counter allows one to create multiple instance of network name-spaces that otherwise are intended to have the same property. For example one can envision two identical but separate network performance to play concurrently. The instance number serves to disambiguate these cases. The rightmost part of the string identifies the performance, and can in principle be composed of substrings that indicate specific sub-networks of the performance. The ServiceID `Net1urMus` given above observes this convention. Dashes are encouraged as separators, especially if there is substructure or if no instance number is given. For example `dev1-jam-urMus` is a ServiceID denoting a performance jam with a subfunction dev1. No instance number is provided and it addresses the urMus platform.

2.2. Integration with OSC

Open Sound Control [10] is the dominant standard for organizing and transmitting sound control information over networks. It allows to structure a the name-space of musical controls in a way that is suitable for open semantic definition and networked exchange.

One of OSC main advantages is the openess of the architecture. The name-space of OSC can be arbitrary and the semantics of the name-space is, with a few exceptions, not really defined by the standard itself. This also is one of OSC drawbacks. It is not clear what functionality and semantics a provider of an OSC service will offer.

The semantic can be ab initio known. In this case the use can simply be statically programmed. Another proposed solution includes the query or browsing of the name-space [11].

These already hint at multiple paths of integration of OSC with network name-spacing and service discovery. One is to simply provide discovery of a project that does not directly relate to OSC and imply by the connection knowledge about the OSC exchange to be used. For many integrated projects such as scripted performances this is a rather viable route. It has been proposed to make OSC discoverable via `_osc._udp` which allows a base-line discovery of this type [8].

Another possible option is to mirror the OSC name-space in the network discovery name space. Then each individual possible OSC message can be individually discovered and addressed. It is not clear if this always makes sense. Some name-spaces are constructed to offer complex function through an ensemble of entries and an individual name-space is not useful in isolation.

Finally one can integrate OSC with a complex query system, perhaps discover the query mechanism which in turn could be standardized.

We have used a combination of project centric and message exposing mechanisms for integrating OSC. For many projects this was suitable. However it is unlikely that these approaches scale, hence alternative mechanisms may need to be developed. However in all cases we feel that a standardization will be critical. For example there are multiple ways to map OSC name-space into NSID format and this mapping needs to be agreed upon. In our work we have simple replaced the standard OSC path slash (/) with a dash (-) stripping the leading slash.

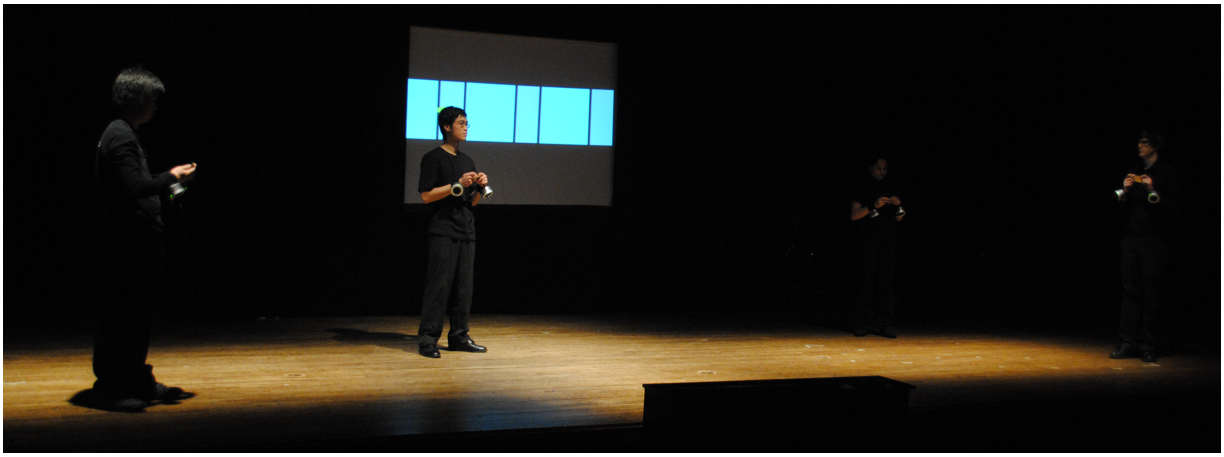

Figure 2. The Michigan Mobile Phone Ensemble performing the networked piece Space Pong by Gayathri Balasubramanian & Lubin Tan, using five iPod Touches (four used by performers, one driving the projector display).

3. BONJOUR IN URMUS

We have implemented these proposals to augment the ease of use of the networking capabilities of urMus [5]. This has two aspects. One is to allow instrument and performance designers access to Bonjour in a flexible yet convenient way, and the other is to allow seamless and automated connection of dataflow pipelines across the network.

3.1. Lua API support of Name-spacing the Network

We implemented simple API functions that allow an urMus lua program to use network service discovery. In order to publish an ongoing advertisement of a namespace, the programmer can use `StartNetAdvertise( ''namespace'', port)`. This is in turn converted into an NSID of the form `_namespaceurMus._udp.`. urMus currently automatically assigns the device name as the human readable string. Once a service has been started it will be discoverable. In order to discover a namespace the `StartNetDiscovery("namespace")` function can be used. This command can be issues independent of whether a namespace already exists or not. If a service with the matching namespace is discovered, and event called `OnNetConnect` is invoked. Should a namespace disappear, because the program is ended or the service as terminated, the event `OnNetDisconnect` is called. The simplest possible full implementation looks as follows:

```
nsr=Region()

local function NewConn(self, ip)
  DPrint("connect "..ip)
end

local function LostConn(self, ip)
  DPrint("disconnect "..ip)
end

nsr:Handle("OnNetConnect", NewConn)
nsr:Handle("OnNetDisconnect", LostConn)

StartNetAdvertise("test",8888)
StartNetDiscovery("test")
```

In this example the namespace `test` on port 8888 is both advertised and discovered on the same device. The implementation does not distinguish between self and other devices and one can choose to listen to ones own service. The ip passed to the events is the network address string of the device that connected or disconnected.

3.2. Automated Connectivity in Dataflow Pipelines

urMus contains a flexible-rate dataflow pipeline [6] primarily for audio processing. One of its primary functions is the support of graphical patching interactions and it exposed to a prototype patching interface of urMus. One of the main goals of the patching environment is to remove the need for any typing interaction. Ideally multitouch motions should be sufficient to generate patches and functional content. Traditionally networking has posed a challenge to this goal. Establishing network connection required some amount of user input, by typing in IP addresses or by selecting from a set of pre-defined addresses. The need to automate this process was one of the driving forces in developing network service discovery-based solutions. The current scheme makes this trivial. When the dataflow uses networking it both advertises network availability and looks for existing networks. If a NetOut is placed it will broadcast to other participants in the network, whereas if a NetIn is placed it will be receiving these broadcasts (Figure 1).

4. USE IN PERFORMANCE

Typically a networked performance simply needs to know which devices participate so it can distribute information between them. Take the performance Space Pong (Figure 2). It uses 5 iPod Touches. Four of the devices are used

by performers and one is used to drive a projected display that augments the performance. The performers can pass a virtual ball around, and the locus of the ball is transmitted over the network. Each player has a subregion in which they can act on the ball and the strokes they perform will change the direction and speed of the ball. One device is set up as a master, and awaits connections of all other devices. Once the connections are established, the master will transmit the ball movements to all clients as well as collect information about strokes that the performer enacted. It is trivial to organize this setup with network service discovery. The master advertises one service and all clients can find the master by looking for it. Once discovered all clients initiate standard OSC connections with the master and start exchanging the needed information.

Scalable ad hoc solutions are also possible. Participatory networking refers to the ability to join and participate in a networked performance without the need to configure or handle invitations. The new participants simply joins the network by launching the musical application or patch for the performance starts to participate. To join a network it is sufficient to discover if the performance already exists and then join it, or set it up if it has not started yet.

5. CONCLUSIONS

In this paper we have discussed how zero configuration networking can be used to generate name-spaced organization of network performances and how this is integrated into urMus. The benefits of this approach is the removal of required manual configuration and the associated maintenance requirement as well as fragility with respect to changes in the network and device availability.

This work integrates well with existing approaches to music control in networks via OSC but standardization will be needed. This standardization is future work. Furthermore we have not addressed service discovery in very long distance networking cases. In principle the approach discussed here does work, but exposure of services very broadly had numerous drawbacks including vulnerabilities to malicious attacks as well as load on the network. Hence other approaches may be more suitable for long range networking.

The current solution allows for ad hoc participatory networking, where performers can join and leave a networked performance without any intervention, and networked performance can use multiple network name-spaces to organize functionality that is robust against other performance setups being present.

6. ACKNOWLEDGEMENTS

Thanks to the student of the University of Michigan couse "Mobile Phones as Musical Instruments" : Gayathri Balasubramania, Yuan Yuan Chen, Chandrika Dattathri, Alejandro Guerrero, Andrew Hayhurst, Kiran Jagadeesh, Steve Joslin, Kyle Kramer, Billy Lau, Michael Musick, Lubin Tan, Edgar Watson. This work was supported in part by a Curriculum Innovation Grant by the Office of Undergraduate Affairs of the College of Engineering of the University of Michigan.

7. REFERENCES

[1] Apple, "Bonjour Printing Specification," 2005. [Online]. Available: http://developer.apple.com/networking/bonjour/-bonjourprinting.pdf

[2] N. J. Bryan, J. Herrera, J. Oh, and G. Wang, "MoMu: A Mobile Music Toolkit," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, Sydney, 2010.

[3] J.-P. Cáceres and C. Chafe, "JackTrip/SoundWIRE Meets Server Farm," *Computer Music Journal*, vol. 34, no. 3, 2010.

[4] A. Cart, P. Rebelo, and A. Renaud, "Networked music performance: State of the art," in *AES Conference: 30th International Conference: Intelligent Audio Environments*, 3 2007.

[5] G. Essl, "UrMus — An Environment for Mobile Instrument Design and Performance," *Proceedings of the International Computer Music Conference*, 2010.

[6] ——, "UrSound — Live Patching of Audio and Multimedia using a Multi-Rate Normed Single-Stream Data Flow Engine," *Proceedings of the International Computer Music Conference*, 2010.

[7] E. Guttman, "Autoconfiguration for IP Networking: Enabling Local Communication," *IEEE Internet Computing*, vol. 5, pp. 81–86, 2001.

[8] R. Muller, "Bonjour/ZeroConf and OSC," July 3 2006. [Online]. Available: http://opensoundcontrol.org/topic/110

[9] J. Oh, J. Herrera, N. J. Bryan, L. Dahl, and G. Wang, "Evolving the Mobile Phone Orchestra," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, Sydney, 2010.

[10] A. Schmeder, A. Freed, and D. Wessel, "Best Practices for Open Sound Control," in *Linux Audio Conference*, Utrecht, NL, 01/05/2010 2010.

[11] V. team, "Minuit : Propositions for a query system over OSC," 2010. [Online]. Available: http://www.plateforme-virage.org/?p=1444

[12] G. Wang, G. Essl, J. Smith, S. Salazar, P. Cook, R. Hamilton, R. Fiebrink, J. Berger, D. Zhu, M. Ljungstrom, A. Berry, J. Wu, T. Kirk, E. Berger, and J. Segal, "Smule = Sonic Media: An Intersection of the Mobile, Musical, and Social," in *Proceedings of the International Computer Music Conference*, Montreal, 2009.