Distributing Mobile Music Applications for Audience Participation Using Mobile Ad-hoc Network (MANET)

Sang Won Lee
Computer Science & Engineering
University of Michigan
2260 Hayward Ave
Ann Arbor, MI 48109-2121
snaglee@umich.edu

Georg Essl
EECS and Music
University of Michigan
2260 Hayward Ave
Ann Arbor, MI 48109-2121
gessl@umich.edu

Z. Morley Mao
EECS Department
University of Michigan
2260 Hayward Ave
Ann Arbor, MI 48109-2121
zmao@umich.edu

Abstract
This work introduces a way to distribute mobile applications using mobile ad-hoc network in the context of audience participation. The goal is to minimize user configuration so that the process is highly accessible for casual smartphone users. The prototype mobile applications utilize WiFiDirect and Service Discovery Protocol to distribute code. With the aid of these two technologies, the prototype system requires no infrastructure and minimum user configuration.

Keywords
audience participation, mobile music, ad-hoc network, network music

1. INTRODUCTION
Today's ubiquity of smartphones and proliferation of mobile applications facilitate new forms of artistic expression emphasizing audience participation. Hence it is important to be able to distribute and manage interactivity of audience in performance setting. In this paper we utilize mobile ad-hoc network (MANET) for distributing mobile applications to a group of people in a particular space so that the distributed application can be used for audience participation.

Mobile ad-hoc network (MANET) consists of voluntary mobile hosts that communicate with each other without a fixed infrastructure in the middle. In MANET, mobile devices are the nodes of the network, and it allows seamless communication at low cost [1]. Unfortunately, MANET has limited scalability so that it is not ideal for general-purpose networking. However, MANET can be useful for applications which incorporate small-scale networks and are tolerant to limited bandwidth. In [2], MANET was used for a text-based personal communication system based on location profiles of users.

Audience participation is one effective way to engage the audience in the contexts of music performance [6], presentation [10], and interactive classroom experience [12]. Often, performers hands out additional devices (e.g. light sticks, clickers, paddles from above examples) to the audience so that they can transmit interaction data to the presenter of the event. The cost of devices is in proportion to the number of participants, and they only allow limited types of interaction (e.g. multiple choices for clickers). An alternative is to utilize personal smartphones of the audience and use mobile applications [3, 8, 9, 11]. However, this forces the presenter to guide the audience to configure their mobile phones (to download an app from an app store, to change network connection to a specific one provided, to type IP address of mobile devices or to type a given session name inside the downloaded app), which creates an additional technical barrier for casual mobile phone users. Web-based approach [13] requires less configuration effort for audience (e.g. launching web browser, typing web address) but user interfaces on the web browser are limited at the moment in a way that it cannot take advantage of native functionalities of a mobile phone (e.g. sound synthesis, sensors).

The use of IP spoofing has been proposed recently for the purpose of content distribution to audience members [7] although it requires that only one network router be present or the audience be instructed how to connect to it. Ad-hoc networking has previously been discussed in the context of structuring networked mobile performance [5]. MANET is more general than ZeroConf networking hence our work expands the scope of the previous work as well.

NIME'14, June 30-July 3, 2014, Goldsmiths, London, U.K.

2. DISTRIBUTION ENVIRONMENTS

2.1. Target System and Data
In this work, we utilize *urMus* [4] which allows flexible design of interactive mobile applications and is able to run a mobile application by transmitting program code over wireless network. A user can access the mobile phone from a computer by typing IP address on a web browser. The mobile phone, as a web server, provides a remote programming environment so that a user can write a program and send it to the mobile phone to run the code within *urMus* application. However, the environment provides no easy method to distribute program code to multiple mobile devices other than to connect each mobile device one by one.

Given *urMus* can run mobile applications by transmitting relatively small sized data (code text); a typical *urMus* program is smaller than 10 KB and rarely exceed 20 KB. It makes suitable for MANET transmit data in fairly short time. Since the application distributed will be determined by code text, it enables a wide range of interactions, from playing a mobile music instrument to answering true/false questions. The central observation why MANET is desirable for our purposes is that no additional infrastructure is required other than mobile devices that audience and the presenter already have. A typical scenario where this system can be useful is a Bring-Your-Own-Device (BYOD) event or any public event where most of people already have smartphones.

2.2. Design Goal
Since this network is utilized to organize audience participatory event in a local setting where the presenter and audience can communicate verbally, high latency is acceptable given that data transmission will only happen intermittently or once during the concert. However it is

problematic if the total time distribution increases in proportion to the size of the audience (e.g. 5 seconds per person in case there are 100 people in the event, which is over 8 minutes). Ideally, the distribution process should not increase linearly in time so that it is scalable in terms of total time of dissemination.

One of the most important requirements of the system is accessibility: it has to be easy to use (network configuration, data transmission). Many systems rely on some type of user configuration (typing IP address, typing URL, downloading an app, changing network to a designated WiFi network, etc.). Those actions are not always intuitive to general audience. We wish to minimize the number of user actions as well as reduce the complexity.

3. CODE DISTRIBUTION METHOD

There are many possible options to realize the goal of distributing data to people, from mobile ad-hoc network to centralized web server structure. In Table 1, we review possible options available today that are able to fulfill similar requirements and compare them in terms of accessibility (user configuration).

Table 1 Possible Networking methods for Distribution

Code Distribution Methods	Pros and Cons
Server–Client/Cloud	+ straight Forward Concept − additional Infrastructure − requires user configuration (no proximity information) − requires connectivity
MANES[1]	+ can simulate proximity + support any types of connectivity (WiFi/Mobile) − need to install one more mobile application
Searching devices over a WiFi network	+ less user configuration − devices needs to be connected to a same network. − limited in searching devices
DNS-based Service Discovery	+ zero configuration in broadcasting /discovering − devices needs to be connected to a same network.
AllJoyn	+ high level API that supports many different platforms − devices needs to be connected to a same network.
WiFi-Direct	+ proximity based connection + wide coverage (no WiFi network /3G/4G needed) − not scalable for a number of nodes − supports only WiFi Direct certified devices.

We chose WiFiDirect and Service Discovery. WiFiDirect is appropriate for the goal of achieving minimum user configuration. WiFi-Direct is a way to enable peer-to-peer communication between mobile devices. It does not require any infrastructure so that the users do not need to change the network configuration of the device (e.g. changing network, typing IP address so on). In addition, WiFiDirect outperforms other similar methods such as NFC or Bluetooth in terms of coverage range or speed. However, one drawback of WiFiDirect is that the network topology can only be one to one or one to a few. The number of devices in the network is expected to be "smaller than the number supported by traditional standalone access points[2]."

Further we utilize Service Discovery Protocol based on WiFiDirect, which is available from Jelly Bean Operating System (Android 4.1 or higher). Service Discovery Protocol (SDP) is network protocols that allow automatic detection of devices and services offered by the devices on a network. Without Service Discovery, a participant needs to choose a device to connect, which is not desirable not only because it will require one extra user configuration but also because a participant needs to know which device will have data to be retrieved among many WiFiDirect enabled devices of other audience members. Using service discovery, information of state of a device such as whether it has (or needs) up-to-date code can be incorporated in broadcasting a service.

[1] http://whispercomm.org/manes/

[2] http://www.wi-fi.org/knowledge-center/faq/how-many-devices-can-connect

4. CODE DISTRIBUTION DESIGN

For a prototype purpose, two mobile applications have been implemented, one for the performer who has data to be distributed (acting as server) and the other for audience who has to receive data (client application). It is built on Android platform. The following sections will cover our design of code distribution procedure.

4.1. Data Transfer Iteration

Forming MANET using WiFiDirect for this purpose is challenging because MANET does not allow many nodes in one connected network. The basic approach taken in this work is to make one-to-one connection multiple times to distribute code and close the connection immediately after data transfer is made. For each connection, the device that has code will find a device that needs the code using SDP, makes the connection, transfer the code using WiFiDirect, close s the connection and look for another peer.

In an actual performance, participants are instructed to download an app from app store (Google Play Store or Apple App Store) and launch the app. Once the app is launched, the process of data transfer begins as stated in Figure 1. When a participant launches the client application, it immediately broadcasts a service. Whether a device of a participant broadcast service or not will be an indicator that the device has up-to-data code or not. Although it is more intuitive to have the server application broadcast a service and the client application to discover the service to retrieve the code, by migrating broadcasting to audience device, we can guarantee exactly one user action per user, which will be discussed in the following section in detail.

In the meantime, the server application tries to discover a service within the range of WiFiDirect covers (with ranges up to 200 meters). Once the server finds a service, it picks one of the devices that broadcasts a service and make connection automatically. Note that there will be services as many as the number of participants and the server application will pick the first device discovered. Once the server application make a connection request to a client application, a participant needs to confirm the pairing by pressing accept button on the screen, which is the only user action for the participant for network configuration. After connection is established, the server application sends data immediately to the client application. When audience application receives data, it sends an acknowledgement message to the server application. The server application closes the connection with the client application upon receipt of acknowledgement. Since the client application received data of interest, it needs not to broadcast a service any more. After closing the connection, the server application tries to discover another peer that broadcast a service again. Naturally, client peers that did not receive data will have a chance to make connection to the server application. Eventually, the server application can iterate the process of making a connection and data transmission to all peers until no peers broadcast services. Note that the only user configuration required in the whole process is for a participant to press "accept" button (other than downliading and launching the app).

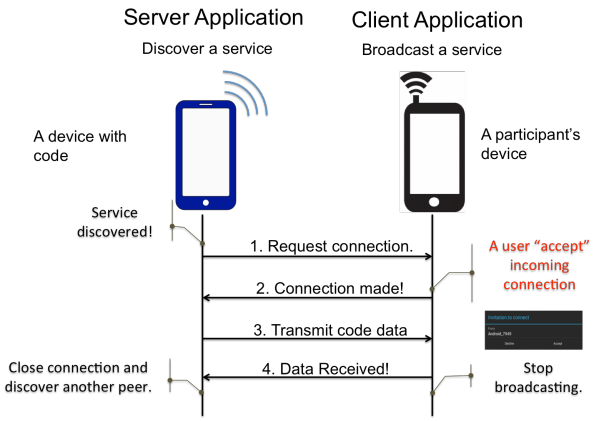

Figure 1 the process of a server application (left) transmitting code data to client application (right) for one audience member.

4.2. Scalable Distribution

Although this iterative connection model described in the previous section accomplishes minimum user configuration, this model is not scalable. If there are *n* peers, the total time is *n* times the duration of data transmission for one peer. The time it takes to transmit data is instant but the time it takes to initiate connection (discover/request/ negotiate) is not. It takes several seconds to initiate a WiFiDirect connection. The total time of data distribution may be reduced if transmissions happen in parallel. However, WiFiDirect is not designed to support multiple devices, it is hard to initiate connection to large group of peers (even 3-4 devices are challenging). To expedite the process, the client application will distribute the code as well once it retrieves the code successfully. As soon as a device received code, it will look for a device that did not receive the code. In other words, the client application with valid data will stop broadcasting and try to discover other client application that is broadcasting a service to seek for code distribution. Initially the only peer, which has code data, is the server application, but once it transmits code to a peer, the peer and the server application are able to transmit code to two other client applications. The number of devices that can distribute data then grows exponentially. Figure 2 shows the example of the whole process in four iterations when there are 15 devices from the audience and one device that runs the server application. The time it takes to distribute to whole group will be $O(\log_2 N)$. For example if there are 1000 participants and if it takes 5 seconds to transmit to one peer, it will take $\log_2 1000 (\approx 10)$ times 50 seconds (less than 1 minute) which is much more scalable than one device transmitting to all audience devices, which will take 1000 times five seconds. In addition, since this network is *weakly connected* network where devices are not always connected but share its state by service discovery, it is completely fine for a node to join and leave the network.

One important characteristic in the basic connection model described in 5.1 is that client applications broadcast a service and the server application makes a connection request automatically when it discovers the service, which seems counter-intuitive. Recall that the device that receives the connection request will require a user to press "accept" button. In this way, we can guarantee only one button press per user, otherwise, some participants will have to keep pressing "accept" button many times, which will interrupt the participant's interaction with the device with a modal message dialog.

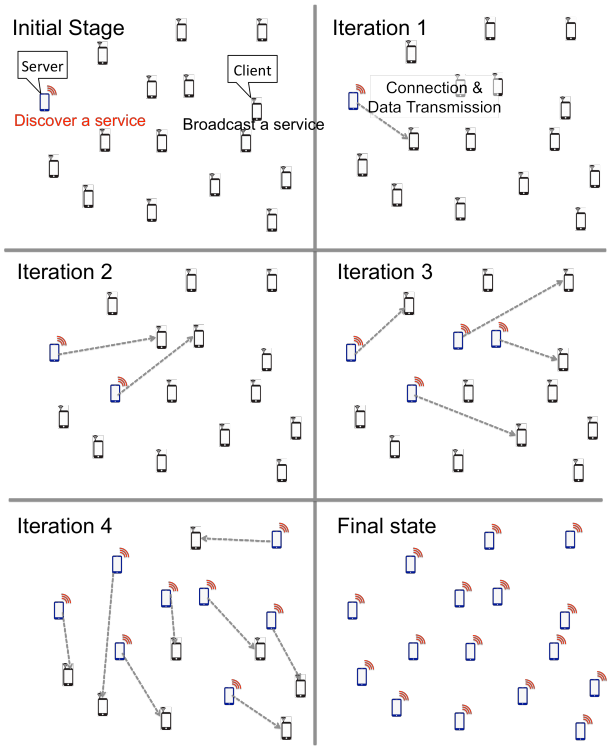

Figure 2 From upper left, it shows how one server application distribute code to 15 different devices. See it takes four iterations of connection procedure to distribute mobile application code.

4.3. Updating Mobile Application

Once all devices received code, all devices will try to discover a service but none will broadcast a service (the final state in Figure 2). This state is problematic in a scenario when the performer wants to distribute more than one application. For instance, imagine there are two audience participation pieces at a performance and the performer wants to change the musical instruments for each music piece. Or if he or she wants to update the nature of the musical instrument during a music piece so that it will change the texture of the overall sound. In this case, the final state (from Figure 2) where all devices try to discover service makes it impossible to distribute the code because server application can only distribute code to the devices in broadcast mode. To distribute the new code to all audience again, the network have to go back to the initial state.

The approach we took to restore the initial state in this work is to add a property of *version* in each service discovery and service broadcast. Service Discovery Protocol allows a programmer to add text data and any customized information within the size constraint (less than 200 bytes). We embedded the version of the app in the service so that both *discovery mode* and *broadcast mode* is associated with an integer number that indicates the version of the application to be distributed. The modified procedure is exactly same with the previous one except that the code will be distributed to the device in broadcast mode only when the version is less than or equal to the version of the device in discovery mode. In addition, if a client application in discovery mode discover a service that has version greater than the number it tries to discover, the application is programmed to go back to broadcast mode with updated version. This state transition happens much faster than the code distribution procedure because the client application only discovers broadcasted service and does not need to make connection with other

devices at all. In addition, the signal that new distribution will be distributed go viral. If one of the client application discover a service with greater version, it will go to broadcast mode immediately with the updated version number which again help other devices to discover the service and go back to the broadcasting mode. This will change the whole audience to go back to broadcast mode quickly again in exponential manner. A few seconds later, the whole group will reach the initial state and the server application can again discover a service with version 2 like it did in the initial state.

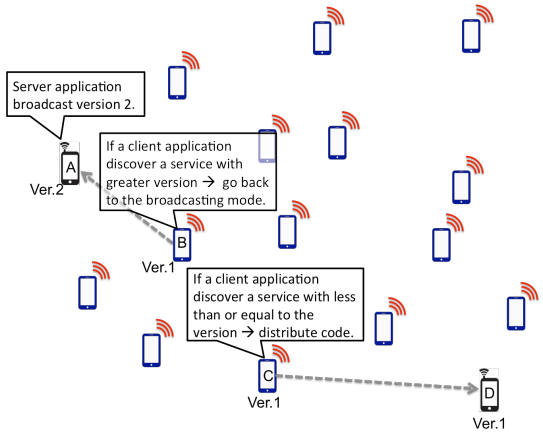

Figure 3 The server application (A) broadcast a service to make a device (B) go back to broadcast mode. Note that the device that discovered the server application broadcasted service (B) will again broadcast a service with version 2 and signal other devices to go back to the broadcast mode. The connection between (C) and (D) is usual scenario that were modified with version comparison so as to run the usual code distribution described in section 5.2.

5. PROTOTYPE DEVELOPMENT

The server application is implemented as a simple mobile application which contains code data and basic functionalities such as discover / refresh / disconnect and broadcast with version data. It prints out the current state of ongoing connection and discovered services for monitoring purpose. The client application is implemented on top of urMus application. It does not have any user interface regarding the code distribution other than a simple message on launching screen which instructs to wait for code distribution and accept incoming connection if any. The client application does not run anything at all if there is no server application that distributes the code.

We tested two applications to evaluate code distribution in a lap setting. The system serves its function of distributing mobile application successfully[3]. As designed, the code distribution procedure is efficient in achieving the goal of making it accessible to use. It requires no configuration other than downloading the client application from app store, launching the application and pressing a button to accept the incoming connection. The time it takes to make one connection between a pair of device is measured from multiple trials. Most of the times, the iteration is complete within nine seconds at most. Analytically, we can calculate the estimated time of completion for large-scale audience. For example, 90 seconds will allow ten iterations of connection and cover about 1000 devices ($2^{10} = 1024$).

[3] Demo video is available at http://youtu.be/wGTe5w7DRl0

6. CONCLUSION

This work introduces a way to distribute mobile applications using mobile ad-hoc network in context of audience participation. It utilizes WiFiDirect and Service Discovery Protocol to distribute code. With the aid of these two technologies, the system requires no infrastructure and minimum user configuration in the procedure. It also succeeds in mobile application distribution in a batch with acceptable latency for a music performance. Currently we have tested this system in small-scale but not yet in a real music performance. We plan to make improvement in both performance (time) and reliability on this method and realize an audience particiaption music piece in the near future.

7. REFERENCES

[1] S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic. *Mobile Ad Hoc Networking*: Wiley. com, 2004.

[2] D. R. Bild, Y. Liu, R. P. Dick, Z. M. Mao, and D. S. Wallach. Using Predictable Mobility Patterns to Support Scalable and Secure Manets of Handheld Devices. in *Proceedings of the sixth international workshop on MobiArch*, 2011, pp. 13-18.

[3] L. Dahl, J. Herrera, and C. Wilkerson. Tweetdreams: Making Music with the Audience and the World Using Real-Time Twitter Data. *the International Conference on New Interfaces for Musical Expression (NIME)*, Oslo, Norway, 2011.

[4] G. Essl. Urmus - an Environment for Mobile Instrument Design and Performance. *International Computer Music Conference*, New York, 2010, pp. 76-81.

[5] G. Essl. Automated Ad Hoc Networking for Mobile and Hybrid Music Performance. *International Computer Music Conference*, 2011, pp. 399-402.

[6] J. Freeman. Extreme Sight-Reading, Mediated Expression, and Audience Participation: Real-Time Music Notation in Live Performance, *Computer Music Journal,* vol. 32, pp. 25-41, 2008.

[7] A. Hindle. Swarmed: Captive Portals, Mobile Devices, and Audience Participation in Multi-User Music Performance. *International Conference on New Interfaces for Musical Expression (NIME)*, Daejeon, South Korea, 2013.

[8] H. Kim. Moori: Interactive Audience Participatory Audio-Visual Performance. in *ACM, Creativity and Cognition*, Atlanta, USA, 2011, pp. 437-438.

[9] S. W. Lee and J. Freeman. Echobo : A Mobile Music Instrument Designed for Audience to Play. *the International Conference on New Interfaces for Musical Expression (NIME)*, Daejeon, South Korea, 2013.

[10] D. Maynes-Aminzade, R. Pausch, and S. Seitz. Techniques for Interactive Audience Participation. *IEEE International Conference on Multimodal Interfaces (ICMI)*, Pittsburgh, PA, 2002, p. 15.

[11] C. Roberts and T. Hollerer. Composition for Conductor and Audience: New Uses for Mobile Devices in the Concert Hall. *the 24th annual ACM symposium adjunct on User interface software and technology*, 2011, pp. 65-66.

[12] J. R. Stowell and J. M. Nelson. Benefits of Electronic Audience Response Systems on Student Participation, Learning, and Emotion, *Teaching of psychology,* vol. 34, pp. 253-258, 2007.

[13] N. Weitzner, J. Freeman, Y.-L. Chen, and S. Garrett. Massmobile: Towards a Flexible Framework for Large-Scale Participatory Collaborations in Live Performances, *Organised Sound,* vol. 18, pp. 30-42, 2013.