PhoneNet: a Phone-to-Phone Network for Group Communication within an Administrative Domain

Te-Yuan Huang Kok-Kiong Yap Ben Dodson Monica S. Lam Nick McKeown
Stanford University
{huangty,yapkke,bjdodson,lam,nickm}@stanford.edu

ABSTRACT

This paper proposes PhoneNet, an application framework to support direct group communication among phones without relay nodes. PhoneNet presents the familiar abstraction of a multi-user chat service to application writers. It performs two main functions: inviting participants to the chat room and routing data between participants directly without going through any intermediaries.

Made possible by a generic chat room service embedded in the network itself, all application-specific code in PhoneNet applications runs on the phones themselves. Unlike the conventional server-client model, this design does not require scalable central servers that can handle all simultaneous interactions.

As a first step, we have created a prototype of PhoneNet that works within an administrative domain. The multicast functionality among phones is implemented on top of a software-defined network (SDN). We have developed two applications using PhoneNet: teleconferencing and photo-sharing. Our experience suggests that it is easy to develop PhoneNet applications and PhoneNet appears to be effective in reducing network traffic.

Categories and Subject Descriptors

C2.4 [**Computer-Communication Networks**]: Distributed Systems

General Terms

Design, Experimentation, Performance

Keywords

Software-Defined Networks, XMPP, OpenFlow, NOX, Phone-to-Phone

MobiHeld 2010, August 30, 2010, New Delhi, India.

1. INTRODUCTION

Smart phones have made it possible for individuals to share their experiences and communicate with each other using multimedia in real time. Imagine attending a conference with multiple simultaneous group sessions. We can now drop in virtually on the different talks to decide on the one to attend. We may wish to attend one, while keeping an eye on another. Or, an undergraduate may want to check out the various parties his friends are attending on campus.

It is challenging to implement this kind of multi-party applications in a server-client model, which is the *de facto* standard for multi-party applications today. The challenges arise from the need to maintain the quality of the multimedia stream, to deliver the information in real time, and to support a large number of simultaneous conversations.

An alternative to server-client architecture is the peer-to-peer architecture, which is used, for example, in the very successful and popular VoIP service, Skype. Phones however, due to their mobility and limited power capacity, do not serve well as relays for group chat. Thus, conference calls between phones typically go through PCs as relays.

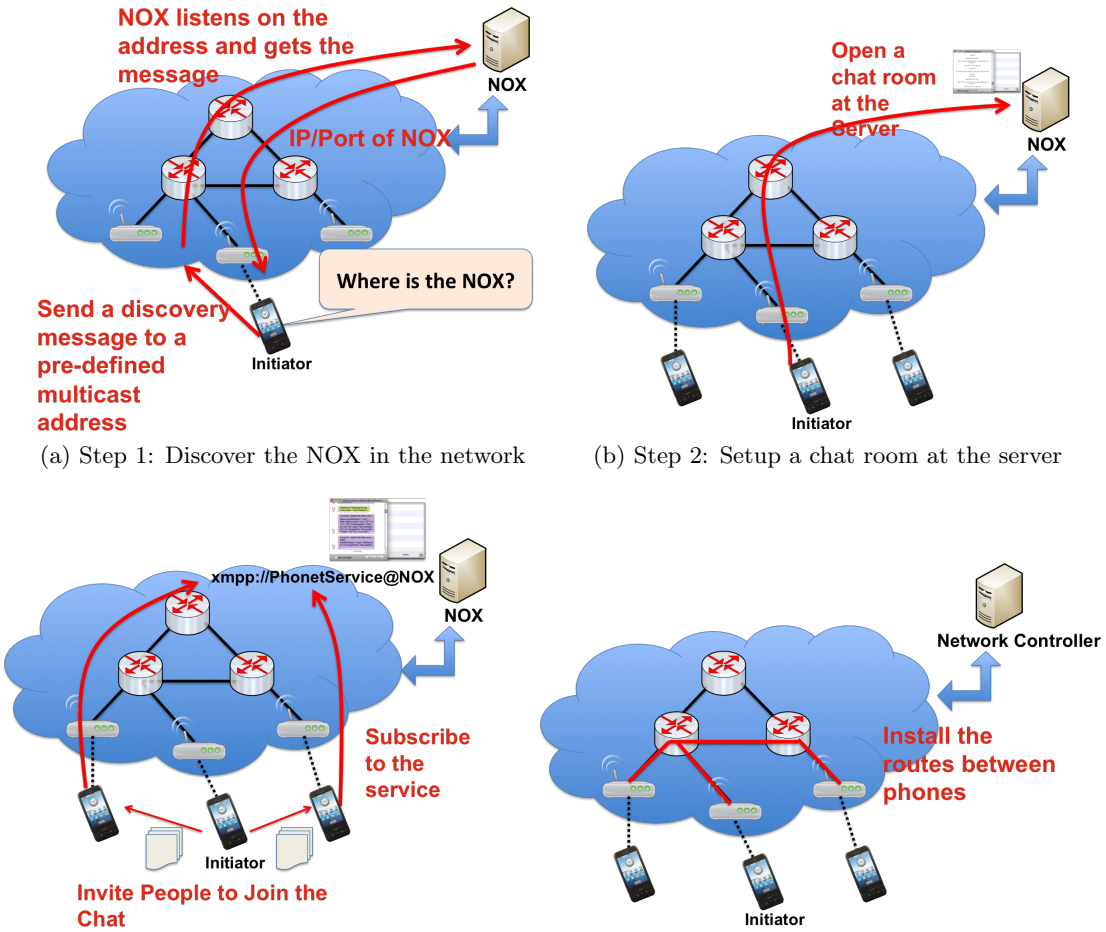
1.1 Direct Communication

It would be ideal if we can eliminate any central server or relay PC, and let the network multicast the many simultaneous real-time streams of high-resolution multimedia data directly. By eliminating the intermediaries, we can reduce traffic in the network, improve the quality, lower the latency, and improve interactivity.

As a first step, this paper focuses on supporting group communication among phones within an intranet on a campus or in an event like a conference. We propose the abstraction of a phone-to-phone network called PhoneNet. In this model, a phone-to-phone network is set up by having all participants join a chat room facilitating the communication. All subsequent messages are routed to all participants directly without going through any intermediary servers.

In this model, all application-specific logic runs on phones directly. Note that the same chat room provider can service any multi-party application; conversely, a multi-party session can be hosted by any chat room server. This eliminates the need for each multi-party application to provide a

This research is supported in part by the NSF POMI (Programmable Open Mobile Internet) 2020 Expedition Grant 0832820, Stanford Clean Slate Program, Google, Xilinx, Cisco, NEC, Deutsche Telekom, DoCoMo and the Mr. and Mrs. Chun Chiu Stanford Graduate Fellowship. The authors would also like to thank Google for its donation of android phones.



(a) Step 1: Discover the NOX in the network

(b) Step 2: Setup a chat room at the server

(c) Step 3: Invite others to join the network

(d) Step 4: Install the phone-to-phone network

Figure 1: Steps to setup a phone-to-phone network in PhoneNet

central service—one that needs to scale up to handle all the simultaneous sessions.

We delegate all the data exchange to the network, eliminating the need for relays, by exploiting the emerging Software Defined Networking (SDN) [4, 8]. More specifically, we utilize the OpenFlow [11] protocol to communicate with switches in PhoneNet and have NOX [5] as the network operating system to control how the packets route through the network.

1.2 Contributions

This paper makes the following contributions:

PhoneNet is an application framework that enables pure phone-to-phone multi-party multimedia applications. We especially focus on better supporting applications with high bit-rate requirements. PhoneNet eliminates the need for relay nodes, which in turn allows resource constrained phones to communicate as a group efficiently.

Implementation. We have implemented a fully functional prototype of PhoneNet, using the XMPP protocol for the chat room service, and using the OpenFlow protocol and NOX to support multicasting in the network.

Case Studies. We have implemented two applications—teleconferencing and photo sharing. Our experience suggests that PhoneNet simplifies programming, and the model greatly reduces the traffic in the network as well as the latency.

We first present the design of PhoneNet in Section 2. We then discuss two applications we have implemented using PhoneNet and investigate how they compare to the current state of the art in Section 3. Finally, we discuss related work (Section 4) before concluding (Section 5).

2. THE DESIGN OF PHONENET

PhoneNet is a framework for configuring and running multi-party applications. It has a control plane and a data plane. The control plane establishes the parties in the group communication and sets up the data plane for direct communication between the phones. The former is shown in Figures 1(a)-1(c); the latter in Figures 1(d). PhoneNet hides all these details from the application writers, allowing them to concentrate on the application-specific logic.

2.1 Establishing the Participants

We build the control plane on top of the Junction framework [6], which is a distributed software architecture aiming

Network controller discovery message

```
{"type": whereisnox"}
```

Network controller discovery reply

```
{"type": "whereisnox",
"tcp port": 2703,
"ssl port": 2703,
"ip": [10.79.1.105,192.168.0.1,172.24.74.198]}
```

Figure 2: A discovery message sent from a user and the reply message from NOX

to enable ad hoc interactions between devices. In Junction, interacting devices rely only on a universal, generic *switchboard* for routing messages. The current implementation of Junction is built on top of XMPP (Extensible Messaging and Presence Protocol), formerly known as Jabber. XMPP is a popular instant messaging (IM) protocol used by many IM software, such as Google Talk and Apple iChat. It is a well-established protocol with several freely available implementations in different languages. In order for participants to meet up with each other, an addressable chat room is set up on the switchboard and used as a rendezvous service. The chat room is implemented using the multi-user chat extension in XMPP.

In PhoneNet, we run the switchboard alongside NOX. In order to establish participants, one of the participants would act as an initiator and perform the following three steps: (1) discover the NOX in the network, (2) set up a chat room on the NOX, and (3) invite other participants and have them subscribe to this group communication by sending service subscriptions to the chat room.

(1) Discover the NOX controller. The communication initiator's first step is to discover the NOX controller, the network operating system, in the network. Similar to the discovery mechanism in DHCP [3], the initiator sends a discovery message to a pre-defined multicast address and port, here being 224.0.0.3:2209. The packets destined for this multicast address and port will be redirected to NOX. Upon receiving the discovery message, NOX would reply with its list of ports and IP addresses, since it might be connected to more than one network (Figure 1(a)). The initiator would then decide which IP address is reachable based on its own address. An example of a pair of discovery and reply messages is shown in Figure 2.

(2) Set up a chat room. After locating the NOX controller, the initiator creates a multi-user chat service (a.k.a. chat room), as shown in Figure 1(b). The chat room is identified by a URI (Uniform Resource Identifier).

(3) Invite participation. The initiator then invites the participants by sharing the chat room URI with them; participants can accept the invitation by subscribing to the chat room (Figure 1(c)). The subscriptions are expressed in JSON and an example is shown in Figure 3.

The initiator can share the chat room URI using a variety of methods [6]. For example, participants may share an application via SMS, email, or IM. This works well for connecting few participants who are already in each others' address book. To take advantage of proximity, the initiator can instead present a two dimensional barcode to other users. We make use of QR codes to achieve this, which provides a visual channel that gives reasonable privacy for

```
{"actorID":"afa907b8-0b9b-442a-8a6e-a4ce8426aeb1",
 "port":[8081,8082],
 "action":"join",
 "IP":"10.79.1.132",
 "multicastAddr":"225.100.100.100",
 "type":"ServiceRequest",
 "jx":{"targetRole":"ServiceBot"},
 "MAC":"00:18:41:dc:4c:d6"}
```

Figure 3: A subscription to join a phone-to-phone network identified by a multicast address.

session creation. A third technique is to broadcast the application details using a radio beacon. We do so by setting the session details as the broadcast name of a bluetooth device. With this beacon, many participants can join a local session at the same time. In the extreme case where all the users of an application interact with each other, there is only one chat room URI, which can be hardcoded into the application.

2.2 Setting Up the Data Plane

We use the OpenFlow protocol and NOX to manage our data plane. OpenFlow [9, 11], supported by multiple vendors, provides an open API and a uniform interface for communicating with switches. In PhoneNet, we use NOX [5] as the network operating system to control and manage the network through the OpenFlow protocol.

(4) Establish a Multicast Session. In order to easily collect subscriptions from the invitees, the initiator installs a service bot as a participant in the chat room. Yap et al. provide a set of APIs for applications to communicates with NOX [13, 14]. The service bot would then digest the subscriptions and inform NOX using the APIs. As NOX receives the subscriptions, it then installs a multicast route between the participants (Figure 1(d)) and utilizes a multicast address to represent this personal phone-to-phone network. We can easily impose further access control by controlling the approvals of subscriptions, while access control is difficult in traditional multicast [2]. Note that participants are allowed to join in even after the session has started. Whenever a new participant subscribes to the chat room, NOX would be notified and then install updated routes. Once the routes are installed, the participants can communicate with one another directly. The PhoneNet uses a multicast IP to represent each group, and the participants would use this address to send and receive messages from one another.

3. CASE STUDIES

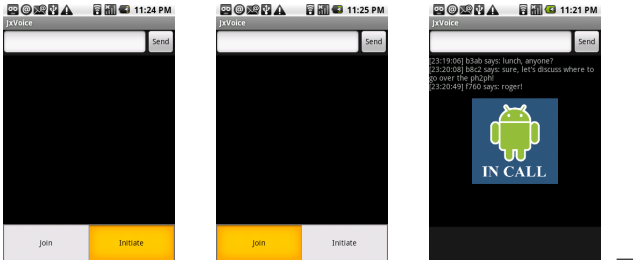
To demonstrate and quantify the utility of PhoneNet, we have implemented two applications. Our experience shows that we are able to achieve better performance and efficiency using PhoneNet as compared to current approaches.

3.1 Teleconferencing

3.1.1 Application Description

The implementation of the teleconference application is relatively simple as all the complexities in setting up the multicast as discussed in Section 2 are taken care of by the PhoneNet framework implementation.

For a user to start a teleconference, the user simply brings up the application and pushes the "initiate" button, as shown



(a) The initiator initiates a chatroom

(b) A invitee joins the chatroom and subscribes to the phone-to-phone network

(c) Once the network is installed, the participants can talk and text with each other directly

Figure 4: Screenshot of the Teleconferencing Application

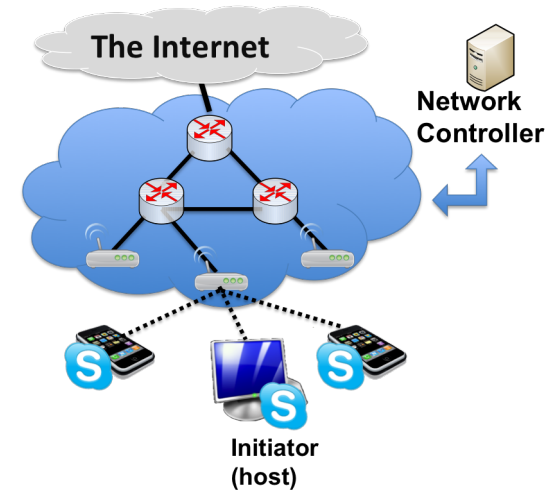
in Figure 4(a). The application would use PhoneNet to discover NOX, setup a chat room, and then asks how the user wishes to invite his friends to participate. The user may choose to invite his friends via SMS or email, at which point, the user's contact information comes up prompting the user to pick the invitees. Upon receiving the invitation, a invitee simply pushes the "join" button (Figure 4(b)) to send a subscription to the chat room. After PhoneNet sets up the multicast route, the users would participate in the teleconference immediately, as shown in Figure 4(c).

3.1.2 *Performance Evaluation*

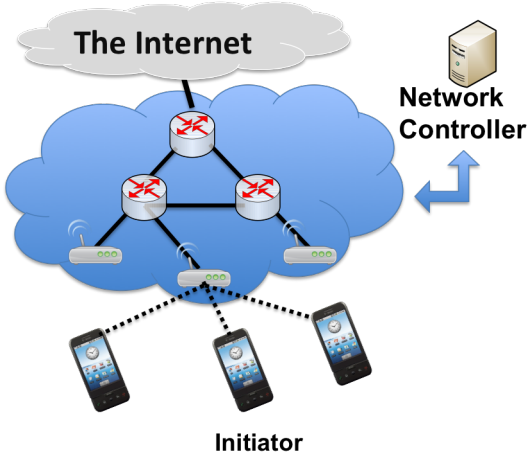
We now evaluate the performance of our teleconferencing application and compare it with the voice conferencing component in Skype. Figure 5 shows the experimental setup for Skype and our application.

We first describe how voice conferencing works in Skype. Skype has released only one handheld version and the platform it runs on is the iPhone. However, it does not allow its iPhone version to host a conference call. Thus in our experiment, we made a three-way conference call between two iPhones and a PC, which also serves as the host of the conference call, as shown in Figure 5(a). We used tcpdump and collected traffic traces at the wireless AP, to which all the three devices connected. We analyzed the trace and plotted the transmission bitrate of each device in Figure 6. From the figure, we find that the trasmission bitrate of the host (the PC) is almost twice as high as the iPhones. This is because the host PC also acts as the relay between the two iPhones and therefore needs to receive and forward the voice data between the iPhones. This explains why Skype does not allow an iPhone to act as a host, since the transmission requirement of a host is much higher than an ordinary participant in a conference call. As the number of participants increases, the transmission bitrate of the host would also increase linearly.

We performed a similar experiment, this time with three Android phones, and collected the traffic trace for the teleconferencing application over PhoneNet. The experimental set up is shown in Figure 5(b). Note that in PhoneNet, we can perform a conference call between three phones and therefore no PC is needed in the experiment. We collected the traffic trace at the wireless AP, to which all the three



(a) Teleconferencing over Skype



(b) Teleconferencing Over PhoneNet

Figure 5: The Experimental Setup for the Two Teleconferencing Systems

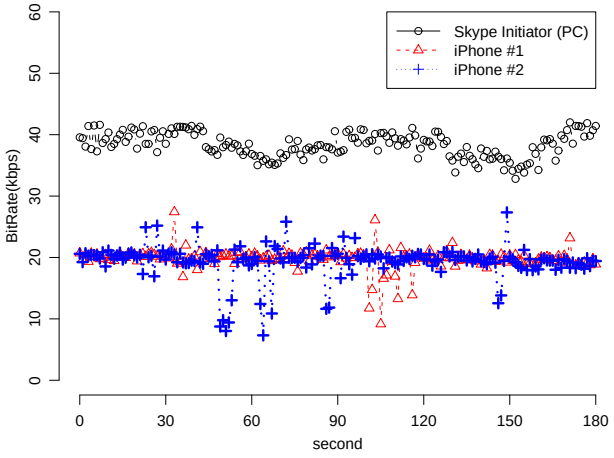


Figure 6: The comparison of the transmission bitrate between the PC (call host) and the iPhones

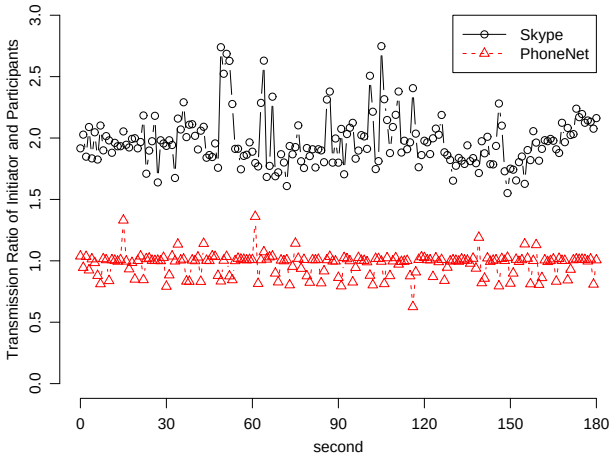
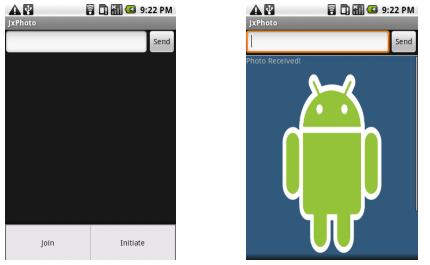


Figure 7: The transmission ratio between the initiator and the participants



(a) The Invite and Join interface

(b) Receives a picture from the sender

Figure 8: Screenshot of the Photo Sharing Application

devices connected. We then compared the ratio of transmission bitrate between the call initiator and the rest of the participants in Figure 7. From the figure, we can see that the call initiator in Skype needs to transmit twice as much data as others in a three-way conference call; while in our application, the call initiator transmits no more data than others. This is because in PhoneNet, we delegate the traffic multicasting to the underlying network and no device needs to act as a relay for others. This is particularly important in phone-to-phone communication because of the limited battery life and bandwidth available to a phone. Furthermore, by eliminating the relay node, the round-trip delay between a pair of phones decreases from 5 ms to 1.6 ms in our experiment setup. The decrease in delay could potentially improve the user experience, especially for multimedia applications.

3.2 Photo Sharing

3.2.1 *Application Description*

Similar to the teleconferencing application, for a user to share photos, the photo sharing application establishes the interested parties and sets up the multicast route through PhoneNet. The application then packetizes the photographs into UDP datagrams and sends them to the participants. If the network is lossless, the initiator only sends out one

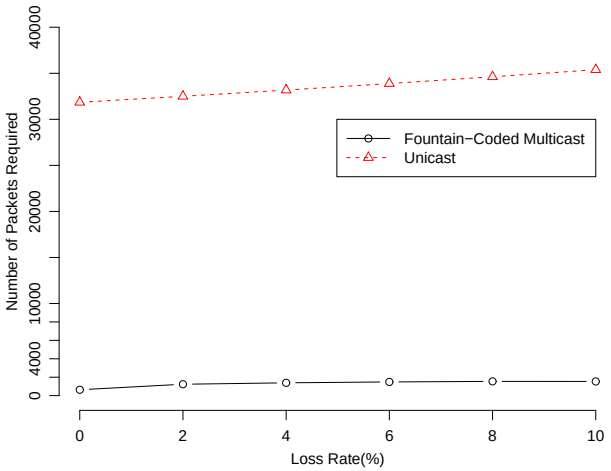


Figure 9: Number of packet transmission required to deliver a 891KB photo to 50 devices.

copy, since packets are duplicated in the network to multiple receivers.

On the other hand, to share photographs or other documents, the application needs to provide reliable transmission over multicast even in a lossy network. Instead of keeping track of packets received by individual receivers, our photo sharing application uses fountain codes (specifically LT codes [7]) to deal with the *cry-baby* problem in multicast. "Cry babies" refer to receivers with lossy channels. With fountain codes, a "cry baby" can reconstruct the photographs with high probability using any of the coded packet received, as long as a sufficient number of packets have ben received. Thus, our application can deliver photographs to multiple parties by continuously sending coded packets to the multicast address until all receivers have decoded the photographs.

3.2.2 *Performance Evaluation*

To evaluate the performance of photo sharing using PhoneNet, we compare it with individual unicasts. The task in our experiment is to share a 891 KB file, delivered using 1414-byte UDP datagrams to 50 other mobile devices.

Since we did not have sufficient phones at the time of the experiment, we simulated the number of packets needed for the fountain-coded multicast algorithm in our photo-sharing application and compared it with the theoretical averages of unicast sessions.

If the network is reliable, the multicast support in PhoneNet reduces the number of packets transmitted by 50 times (Figure 9). This translates to significant savings in energy consumption. Our implementation needs only to keep track of whether a receiver has decoded the photographs; the state of the algorithm is much smaller than unicast implementations, which have to remember the packets received for each receiver. As the loss rate of the network increases, our fountain-coded multicast implementation continues to outperform unicast by over 20 times.

4. RELATED WORK

The Internet Group Management Protocol (IGMP) [1] is a communication protocol used to manage multicast group

membership between a host and its directly attached router in IPv4 system. IP hosts who want to join a multicast group need to inform their neighboring router through IGMP. Between multicast routers, multicast routing algorithms, such as PIM and MOSPF, are used to coordinate the routers throughout the Internet. However, the current Internet multicast service is a "open service model". A host can join a multicast group by simply sending a IGMP "Host Membership Report" message to its router. The sender does not have the control over who can receive the datagrams sent to that group. Similarly, there is no control over who can send datagrams to the group members. A host can even send to a multicast group without joining the group. Worse, senders cannot reserve addresses or prevent another sender from using the same address. The unmanageability of a conventional IP-multicast group results in very limited number of multicast applications. Diot et al. provides a detailed summary on why the IP multicast is not widely deployed in the Internet [2].

In PhoneNet, since multicast routes are only installed between the group members, only the members can send and receive datagrams to/from the multicast address. The initiator can choose with whom it shares the invitation (the chat room URI). Further access control can be imposed by controlling the approvals of subscriptions to the chat room.

5. CONCLUSION AND FUTURE WORK

PhoneNet is an infrastructure which enables group communication between phones within an administrative domain. Any phone user can act as a group initiator and create its own phone-to-phone network through PhoneNet. With the help of software-defined networks, PhoneNet then delegates data exchange directly to the underlying network. Therefore, none of the participants would need to relay traffic for others, which is expensive on phones given limited bandwidth and power constraints. We implemented two applications and show the usefulness of PhoneNet.

We plan to deploy PhoneNet in the production OpenFlow network in our department building [16] and allow users in the building to form their own phone-to-phone network with each other. In our previous works [15], we also explored how to provide lossless handover in software-defined networks for mobile devices with multiple wireless interfaces. Given the mobile nature of phones and the increasing number of wireless interfaces on each phone, we would like to integrate the mobility support into PhoneNet. PhoneNet currently only work inside a single domain, we also plan to extend the system to work between multiple OpenFlow/NOX netowrks. We will have the opportunity to test a large-scale deployment in a couple of years, when OpenFlow-enabled equipment in numerous universities in US and Europe are deployed [10, 12].

6. REFERENCES

[1] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan. RFC 3376: Internet group management protocol. `http://tools.ietf.org/html/rfc3376`.

[2] C. Diot, B. Neil, B. Lyles, H. Kassem, B. N. Levine, and D. Balensiefen. Deployment issues for the ip multicast service and architecture. *IEEE Network*, pages 78–88, January 2000.

[3] R. Droms. RFC 2131: Dynamic Host Configuration Protocol. `http://tools.ietf.org/html/rfc2131`.

[4] K. Greene. Software-Defined Networking. `http://www.technologyreview.com/read_article.aspx?ch=specialsections&sc=tr10&id=22120`, March/April 2009.

[5] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. NOX: Towards and operating system for networks. In *ACM SIGCOMM CCR*, July 2008.

[6] Junction Documentation for Application Developers. `http://mobisocial.stanford.edu/junction`.

[7] M. Luby. LT Codes. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002.

[8] N. McKeown. Keynote talk: Software-defined networking. In *IEEE INFOCOM*, April 2009.

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, April 2008.

[10] OpenFlow in Europe: Linking Infrastructure and Applications. `http://www.ict-fireworks.eu/fileadmin/events/FIA_Valencia/Demeester.pdf`.

[11] The OpenFlow Switch Consortium. `http://www.openflowswitch.org`.

[12] OpenFlow Meso-Scale Deployment. `http://groups.geni.net/geni/wiki/SpiralTwo`.

[13] SFNet: Software Friendly Network. `http://openflowswitch.org/wk/index.php/SFNet`.

[14] K.-K. Yap, T.-Y. Huang, B. Dodson, M. S. Lam, and N. McKeown. Towards software friendly networks. In *ACM APSys2010: First ACM Asia-Pacific Workshop on Systems (in conjunction with SIGCOMM)*, August 2010.

[15] K.-K. Yap, T.-Y. Huang, M. Kobayashi, M. Chan, R. Sherwood, G. Parulkar, and N. McKeown. Lossless handover with n-casting between wifi-wimax on openroads. In *ACM MOBICOM Demo Session*, September 2009.

[16] K.-K. Yap, M. Kobayashi, D. Underhill, S. Seetharaman, P. Kazemian, and N. McKeown. The stanford openroads deployment. In *WiNTECH*, September 2009.