Proximiter: Enabling Mobile Proximity-Based Content Sharing on Portable Devices

Bo Xing[1], Karim Seada[2], and Nalini Venkatasubramanian[1]

[1]School of Information and Computer Sciences, University of California, Irvine. {bxing, nalini}@ics.uci.edu
[2]Nokia Research Center, Palo Alto. karim.seada@nokia.com

I. Introduction

The recent years have witnessed an explosive amount of user-generated contents, along with users' soaring demands on sharing these contents. An evidence of that is the blossom of hundreds of sharing sites on the web, among which the most famous are MySpace [1], Flickr [2], YouTube [3], Facebook [4] and LinkedIn [5]. Through these sites, users upload the contents they generate (blogs, pictures, videos, resumes, etc.), and make them available to their friends or the general public. A common characteristic of such *Internet-based content sharing* scenarios is that all contents are hosted by third-party servers on the Internet, and sharing happens when content consumers (regardless of where they are) access the hosting web sites.

As a complement to Internet-based content sharing, we envision a new form of sharing, which we call *mobile proximity-based content sharing*: contents generated using a portable device (e.g., smart phone) are thereafter hosted on the device itself; sharing happens when content consumers come into the proximity of the content owner (wherever the content owner goes). Imagine that on your first day working in a new company, when you walk around with your cell phone in your pocket, the colleagues you run into may see your profile and your favorite pictures/videos through their phones; rather than just nodding to you, they come greet you and chat with you about anything you might have in common. What a great experience!

Such scenarios are potentially made possible by the pervasiveness of mobile handheld devices and the fast growing capability of these devices. Nowadays a large portion of user-generated multimedia contents (pictures, audios and videos) are actually generated using portable devices with their embedded microphones and high-resolution cameras. While some high-end portable devices now have touch screens and/or QWERTY keyboards, even typing (editing documents or writing blogs) is not as painful as it used to be. Considering the above and the mass storage portable devices now have (up to tens of gigabytes), it is a natural idea to have the contents hosted on the devices where they are generated and have them shared right from there. This notion of "sharing on the move" is further made feasible by the networking technologies that are available on portable devices. Infrared, Bluetooth and Wi-Fi, although varying in communication range, transmission speed and power consumption, are all capable of connecting and sharing data between adjacent devices. In addition, with the multi-hop networking capability of, e.g., Wi-Fi, the notion of proximity can be extended, where devices might not be able to communicate directly but sharing still can take place.

The distinguishing characteristics of mobile proximity-based content sharing makes it very promising. First, since sharing starts at the place where contents are generated right after they are generated, those contents that are location-relevant and/or time-relevant would be more informative to content consumers. Second, as sharing happens directly between portable devices, it does not require Internet access, which is not yet available everywhere. Third, sharing on the move with the proximity fosters local interactions between people and creates opportunities for them to make new friends. It expands one's social network to his/her physical proximity, and then fuses the proximity network back into the social network.

Content sharing between mobile users has recently attracted quite a few research efforts; several interesting applications have been developed. BlueTorrent [6] is a file sharing application designed for mobile users to cooperatively download relatively large files (e.g., advertisements) from stationary Bluetooth access points. MobiUS [7] presents a collaborative video application, where a high resolution video is displayed across the screens of two adjacent mobile devices and users watch the video together on a double-sized screen. The WhozThat application [8] is built around a premise that is similar to one of ours – sharing with the proximity can substantially lower the barriers to social discourse by minimizing unfamiliarity. A basic version of WhozThat shares users' Facebook IDs, while nearby devices that have Internet access can visit the web site and import relevant social context into the local context.

In this work, we design and prototype *Proximiter*, a mobile proximity-based content sharing application. We implement popular content sharing services on portable devices, and show how they are utilized for realizing the idea of "sharing on the move". Our goal is to build a proof-of-concept platform and show the feasibility of this emerging application. Meanwhile, we seek to shed lights on its potential impact on the way people share and interact, and thus stimulate new ideas and perspectives on its design and deployment. In addition, we aim to showcase the use of ad-hoc networking technologies in supporting mobile applications with social networking flavors.

II. *Proximiter*: Application Functionalities

Proximiter enables the sharing of various types of contents on portable devices. Its core functionalities include the following proximity-based services (illustrated in Fig. 1). We identify

Bo Xing was an intern at Nokia Research Center Palo Alto during this work.

978-1-4244-3304-9/09/$25.00 ©2009 IEEE

Fig. 1. *Proximiter*: Application Core Functionalities

their counterparts in the Internet-based services domain to give a better idea of what they offer.

(i) *Text Messaging (Proximity ICQ)*. A user can send text messages to one other user, a group of other users, or all other users in the proximity. This would be useful in conference scenarios where attendees say hello and introduce themselves to others who they are interested in networking with.

(ii) *File Dissemination (Proximity BitTorrent)*. A user can transfer a file to one or several other users in the proximity, or initiates the broadcast of a file. This would be useful in scenarios where, e.g., participants of a meeting need to quickly share meeting documents. In another possible scenario, electronic coupons are distributed by store assistants around the parking lot outside the store to attract customers.

(iii) *Profile Sharing (Proximity LinkedIn)*. A user's electronic business card is viewable by nearby users. This would be helpful for businessmen on travel to get new connections, e.g., at airports or on the flights.

(iv) *Personal Space Sharing (Proximity MySpace)*. A user can create web space (or blogs) on her device, or alternatively the device automatically downloads her personal space from the web when connectivity is available, and nearby users will be able to browse the space through their devices. This would create interesting new topics for company colleagues to talk about and thus encourage more interactions between them.

(v) *Picture Sharing (Proximity Flickr)*. A user, after taking a picture using her device, can request sharing it, and nearby users will be able to view the picture on their devices. If the picture is about a location, this might provide valuable information to people who are approaching that location. For example, through viewing others' shared pictures just taken at a place of interest, tourists would get up-to-date information about the conditions at that place and make better trip planning.

(vi) *Video/Audio Sharing (Proximity YouTube)*. A user, after shooting a video clip using her device, can request sharing it, and nearby users will be able to either download the whole video file or watch the video directly through streaming. This could be a good way for salespersons to advertise their products and promote sales on the move.

(vii) *Video/Audio Conferencing (Proximity Skype)*. Users can perform real-time video/audio conferencing through their devices. Users hence can use the devices as walkie talkies. They may also use them as baby monitors, by placing one device next to a sleeping baby and watching on another device from other rooms. Furthermore, users can collaborate and turn their devices into multi-angle cameras. For example at a sport game, audience sitting at different rows may share their views of the game and get other interesting perspectives as a result.

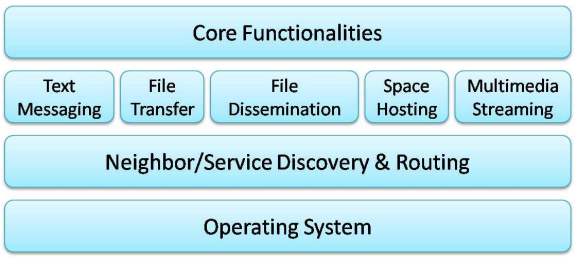

Fig. 2. *Proximiter*: Prototype Implementation Structure

III. *Proximiter*: Prototype Implementation

The prototype of *Proximiter* has been implemented on Nokia N800 Internet Tablets [9], a type of portable appliance that has enhanced wireless networking and media features. Considering factors such as communication range, multi-hop support and beaconing/scanning performance, we use Wi-Fi as the major radio technology, but keep the implementation easily extensible to work with Bluetooth as well. The N800 operating system, Maemo v4.0 (or, OS2008) [10], is a modified version of Debian GNU/Linux, including an X-Windows-based graphical user interface. It allows for setting up Wi-Fi ad-hoc and infrastructure connectivity with configurable parameters (transmission power, power saving mode, etc.). The Maemo development platform provides a GNOME-derived SDK; developments of Maemo applications take place on x86 machines using the Scratchbox Cross Compilation Toolkit.

Proximiter is implemented in a modular fashion, where neighbor/service discovery and routing as the fundamental building block sits at the bottom. On top on that, five modules that provide various middleware services are built. They deal with text messaging, one-to-one file transfer, one-to-all file dissemination, personal space hosting, and multimedia streaming, respectively. Each of the modules contributes to one or several core functionalities of the application. For instance, the File Transfer module is employed by profile sharing as well as picture sharing; the Multimedia Streaming module serves both video/audio sharing and conferencing. Fig. 2 depicts the implementation structure of the prototype.

Neighbor/service discovery and routing is the basis for building all the proximity-based services in *Proximiter*. Neighbor/service discovery is the process of a device detecting who else are in the proximity and what services they provide, whereas routing concerns a device finding a multi-hop path to reach a destination. The Neighbor/Service Discovery and Routing module of *Proximiter* works in both ad-hoc mode and infrastructure mode (two devices connected to the same access point will be detected as one hop away from each other as long as the access point supports WLAN broadcasting), or even in an ad-hoc/infrastructure mixed network. It employs the OLSR (Optimized Link State Routing) standard [11] and accordingly the OLSRD (OLSR Daemon) implementation [12]. OSLR is a proactive routing protocol originally designed for ad-hoc networks. It constructs full routing tables on devices

before actual communications take place and maintains them on a regular basis. This closely fits *Proximiter*'s needs for neighbor discovery. OLSR was selected also because of its efficiency in routing – it avoids blind flooding of control messages by assigning rebroadcasting tasks to a subset of devices called MPRs (Multi-Point Relays). Moreover, owing to the MPR mechanism, OLSR offers excellent support for network-wide broadcast operations, which could be utilized for one-to-all communications in *Proximiter*. Considering that, we have implemented an OLSRD flooding plugin, which takes input data and broadcasts the data in an efficient manner to the whole network. With the flooding plugin, service discovery is simple: a device periodically forwards its available services and the corresponding port numbers to the flooding plugin, which distributes this information to all nodes in the vicinity.

The Text Messaging and the File Transfer modules are straightforward in implementation. The Text Messaging module deals with transmitting text messages. One-to-one and one-to-many text messages are sent through TCP connections between the sender and the destinations, whereas one-to-all messages are delivered through the OLSRD flooding plugin. The File Transfer module is responsible for file transfers between two devices. It simply establishes a TCP connection between the devices, and transfers the file through the connection.

Although the OLSRD flooding plugin can handle network-wide broadcast efficiently, it offers only best-effort service. When it comes to the dissemination of large files, reliability becomes an issue: if a recipient misses any piece of the data, the file is corrupted. To address this problem, we have developed a reliable file dissemination protocol [13]. The source node fragments files into chunks of optimal sizes and generates appropriate metadata that describes the dissemination. The metadata is distributed in the network; proprietary mechanisms are employed to ensure the receptions of the metadata at all receivers. Meanwhile, the actual data is diffused to the receivers, which collaborate in exchanging missing data chunks.

For hosting personal spaces on the devices, we use the Maemo version of HTTPD (Apache Hypertext Transfer Protocol Server) [14], which runs as a standalone daemon process. After a user specifies the directory where the HTML files and other resources are located, HTTPD hosts it on an HTTP server, and listens for incoming connection requests. When handling multiple requests concurrently, it creates a pool of child threads.

Proximiter employs the Gstreamer library [15] for enabling video/audio recording and streaming on the devices. Videos are encoded and decoded using an H.263 codec (provided by the "hantro4200enc" and "hantro4200dec" elements in Gstreamer). Mu-law codec is used for audio encoding/decoding (the "mulawenc" and "mulawdec" elements in Gstreamer). In a one-to-one streaming session, the server runs a Gstreamer pipeline which encapsulates the video (or audio) data in RTP packets and sends them through UDP to a specific (address, port) pair. The client also runs a Gstreamer pipeline listening on that port, which, upon receiving the RTP packets, decodes the payload and displays the video (or plays the audio). Note that a video streaming session typically consists of two parallel Gstreamer pipelines: a video pipeline and an audio

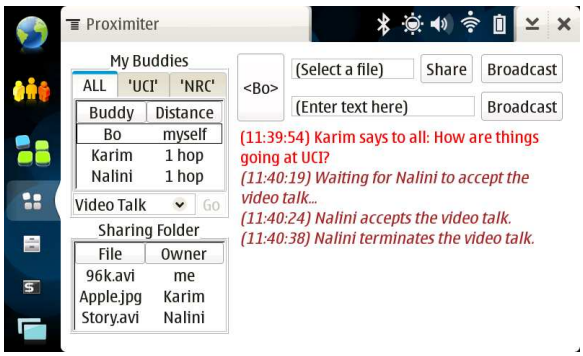

Fig. 3. *Proximiter*: A Screenshot of the Prototype's User Interface

pipeline; synchronization is done at the client end. One-to-many streaming is done by simply directing the server pipeline to multiple (address, port) pairs. Other than one-to-one and one-to-many streaming, we have also implemented one-to-all streaming (real-time video/audio broadcasting): the server pipeline is directed to the OLSRD flooding plugin, and thus the RTP packets are broadcast to all reachable devices.

IV. The Demonstration

In this demonstration, we run the *Proximiter* application on a group of closely located Nokia N800 Internet Tablets, and show how the devices are enabled to share contents on the go. In particular, we demonstrate its core functionalities as described earlier. In the demonstration, the devices can be distributed to interested conference attendees, who can try out the application by themselves. The devices are all configured to work at Wi-Fi ad-hoc mode and to connect to the same SSID (Service Set Identifier). Almost instantaneously after being connected, a device detects the presence of all other devices, and their IDs as well as service availability are displayed in the "buddy list" on the user interface (a screenshot is shown in Fig. 3). Content sharing can then take place when user selects any of the services. If access points are available, we also show that *Proximiter* works in infrastructure mode as well. The demonstration hence has minimum requirements in terms of equipments, network access, space and setup time.

References

[1] http://www.myspace.com.
[2] http://www.flickr.com.
[3] http://www.youtube.com.
[4] http://www.facebook.com.
[5] http://www.linkedin.com.
[6] S. Jung, U. Lee, A. Chang, D. ki Cho, and M. Gerla, "Bluetorrent: Cooperative content sharing for bluetooth users," in *PerCom*, 2007.
[7] G. Shen, Y. Li, and Y. Zhang, "Mobius: Enable together-viewing video experience across two mobile devices," in *MobiSys*, 2007.
[8] A. Beach, M. Gartrell, S. Akkala, J. Elston, J. Kelley, K. Nishimoto, B. Ray, S. Razgulin, K. Sundaresan, B. Surendar, M. Terada, and R. Han, "Whozthat? evolving an ecosystem for context-aware mobile social networks," *IEEE Network*, vol. 22, 2008.
[9] http://tableteer.nokia.com.
[10] http://maemo.org.
[11] T. Clausen and P. Jacquet, "Optimized link state routing protocol (olsr)," 2003, iETF RFC 3626.
[12] http://www.olsr.org.
[13] B. Xing, S. Mehrotra, and N. Venkatasubramanian, "Radcast: Enabling reliability guarantees for content dissemination in ad hoc networks," in *INFOCOM*, 2009.
[14] http://httpd.apache.org.
[15] http://gstreamer.freedesktop.org.