P2P File Sharing on Mobile Phones: Design and Implementation of a Prototype

Claudio E. Palazzi, Armir Bujari, Emanuele Cervi
Dipartimento di Matematica Pura e Applicata,
Università di Padova,
Padova, Italy
cpalazzi@math.unipd.it, abujari@studenti.math.unipd.it, ecervi@studenti.math.unipd.it

***Abstract*—Mobile phones have already evolved from simple voice communication means into a powerful device able to handle multimedia documents, personal productivity applications, and all sort of connections to the Internet. It is hence expected to see a popular application such as file sharing to become widely utilized even in this context. Indeed, the combination among mobile users and file sharing seems a match made in heaven: people could exchange files of interest just passing by each other. Moreover, new context related applications may be developed to exploit this possibility enriching our lives. In this work, we present design issues and implementation details about a real P2P file sharing application for mobile phones.**

Keywords-P2P, Bluetooth, Mobile Phones, File Sharing

I. INTRODUCTION

The mobile revolution is under everybody's eyes: we are all owners of one, or even two, mobile phones and, in some countries, the number of mobile phones have surpassed the number of cabled lines. These devices were initially intended just as voice communication means; now, since their rich functionalities and capabilities, they have deserved the appellative of *smart phones*. They are now able to generate and play any sort of multimedia, they contain a lot of personal information, and we like to personalize them with original background images, themes, and ring tones. As they evolve into a powerful digital assistant, users expect that more and more of those applications usually run on home/office desktops become available also on the screens of their phones.

In the past, mobile phones have had limited memory, limited data communication capabilities, and closed, proprietary operating systems; for these reasons, they have seemed almost immune to file-sharing applications. However, with current memory-intensive smart phones, endowed with several connectivity options (i.e., GPRS, UMTS, Wi-Fi, Bluetooth) and running open software platforms like Symbian and Windows Mobile, peer-to-peer (P2P) systems are ready to colonize even the mobile realm.

Indeed phone users are already exchanging files, pictures, videos, and ring tones through their Bluetooth connectivity. However, this generally happens between two users that know each other very well and decide to transfer some file from one phone to the other after having compared their respective contents.

Instead, the combination of mobile phones with P2P file sharing applications enables the creation of a mobile P2P system in which mobile peers establish peering relationships over wireless links based on proximity. As an example, imagine a user (Alice) whose phone has a certain ring tone A and would like to have a background image B. During her daily activities, Alice walks through the town, takes the bus, enters a store, etc. In the meantime, she passes by a lot of other phone users; so, maybe, Alice will walk at a few meters of distance from Bob, who has image B in his phone or may sit close to Carl on the bus who would like to have the ring tone A. Since their proximity it would be easy for an appropriate file sharing application to use the Bluetooth connectivity of Alices's, Bob's, and Carl's phone to exchange files A and B.

Indeed, current mobile phones have multiple communication technologies built into them (e.g. WiFi, Bluetooth, UMTS) and we could exploit this feature to create a P2P network, which would allow automatic exchange of files and data among phones. However a single device frequently uploading and downloading files using WiFi will quickly become out of power due to battery consumptions. Even UMTS does not represents the best choice as its use is generally associated with a cost. Instead, the best option to guarantee quick connectivity is represented by Bluetooth: it allows up to 1 Mbps of connectivity, it is free, it has a range of about 10 m (hence sufficient for automatic download among customers in a store or commuters on a bus).

Therefore, we have created a proof-of-concept file sharing application for mobile phones that works through Bluetooth connectivity; namely, *P2PBluetooth*.

The first contribution of this paper is that of providing an overview of our solution and practical design considerations to other researchers/practitioners that might be interested in developing a similar project. The second (but not less important) contribution is that of demonstrating a new paradigm of use of P2P solutions that matches file sharing with mobile users, allowing users to exchange files based on proximity to each other, thus fostering new applications.

In particular, the paper is organized as follows. In Section II, we provide some background considerations that may be of help in better understanding the contribution of this work. Section III describes the software platform adopted to develop our application. Our P2PBluetooth is described in Section IV. Finally, Section V concludes this paper and proposes future directions for this work.

II. P2P ON MOBILE PHONES: BACKGROUND

P2P systems in mobile environments face multiple challenges: highly varying online state (*presence*, [1]), hierarchical network structure, and limited device capabilities [2]. Therefore, traditional solutions for P2P over fixed networks may need to be redesigned when applied in a mobile network.

A first example is represented by the fact that mobile phones have capabilities that are much more limited than those of terminals in fixed networks: the latter have more than enough resources, such as processing power, storage capacity, and network bandwidth.

In order to share resources, P2P applications need to support two fundamental coordination and control functions: resource mediation mechanisms (to locate resources or entities), and resources control mechanisms (to permit, update, prioritize, and schedule the access to resources). A pure P2P architecture implements both mechanisms in a fully decentralized manner, whereas hybrid P2P systems utilize central entities, e.g., the eDonkey index servers collect and distribute file location information for all peers [3].

Clearly, resource mediation is more effective when done in a centralized manner, while decentralized resource control could be outweighted by bandwidth costs. Yet, user-provided content and decentralized resource storage are the key features that have made P2P applications successful. It is hence desirable to preserve this feature even when considering a P2P network composed of mobile phones.

In a mobile P2P architecture there are two main restrictions regarding the air interface: a relative low effective bandwidth and high latencies. This makes essential to reduce the signaling overhead as much as possible in order to achieve acceptable performances.

Besides, the limitations of transmission power and battery capacities cause the upload bandwidth to be significantly lower than the download one.

Clearly, during domain analysis some choices have to be made that influence the design and functional behavior of a P2P product. To this aim, in the following subsection, we discuss one of the strategic element for P2P data retrieval: the resource mediation process.

A. Resource Mediation

The process of data search is a key element in data retrieval and a crucial component in the P2P architecture. Trivial searching strategies can undermine the utility of the system and make it useless in most situations. Another feature, common to most P2P software is the ability to share a desired amount of data making it visible externally.

In order to develop a classic P2P system, the presence of a third centralized entity is often necessary (e.g., a database or a centralized list of files that can be shared). This entity has to be visible to all clients and has to provide the correct identification of any file that may be shared. It could be a public service (e.g., a tracker, a web server) whose only job is to manage the incremental update of the list of files made available by the infrastructure. This would guarantee a global and coherent view of the published data to all the clients.

However, a solution of this type adds an overhead to the system as this third entity requires maintenance, whereas it would be desirable to have an entirely independent infrastructure. Moreover, in case of misuse by some users that utilized the infrastructure to illegally share files protected by copyright, the whole (centralized) system may be easily taken down by the authorities. Clearly, it is not our aim to promote piracy: what we are saying is that it would be unfair to have the whole system blocked for a few users' misbehavior. Therefore, we have chosen to utilize a decentralized infrastructure.

To request the download of a specific file, users can specify a keyword. The use of a keyword to identify the data might be trivial because name collisions can occur often but, on the other hand, it is a straightforward solution. Another possible approach is that of indexing the data by computing a digest but in order to avoid potential collisions this job would need to be delegated to a third entity with a global vision of all the files available in the infrastructure.

III. THE ADOPTED SOFTWARE PLATFORM: J2ME

To create a file sharing application for mobile phones, we tried to build a software framework which could be portable in different execution environments; this could be achieved by using the Java technology for mobile devices. J2ME is a software development platform and is currently deployed and supported by a vast majority of mobile telephony vendors. The Mobile Service Architecture (MSA) is an abstract specification defining the platform components, it is a contract specifying the key requirements of a J2ME platform which mobile device vendors should follow.

MSA platform builds on the Java Platform Micro Edition (Java ME) specifications that have come before it, including the Mobile Information Device Profile (MIDP) and Connected Limited Device Configuration (CLDC). The MSA specification define a standard set of application functionality for mobile devices while clarifying interactions between various technologies associated with the MIDP and CLDC specifications. Because there is a lot of variation in mobile handset hardware and software capabilities, the MSA specification implements the predefined subset of the MSA.

P2PBluetooth relies on an MSA subset and the devices on which is installed must support CLDC (JSR 139), MIDP (JSR 118), File & PIM (JSR 75) and Bluetooth (JSR 82) [4].

A. Security Domain

The MIDP 2.0 specification defines an open-ended system of permissions. For instance, to make any type of network connection, a MIDlet must have an appropriate permission. The permissions defined in MIDP 2.0 correspond to network protocols, but the architecture allows optional APIs to define their own permissions.

MIDP 2.0 includes the concept of trusted and untrusted MIDlets.

- An *untrusted* MIDlet suite has limited access to restricted APIs, requiring user approval depending on the security policy of the device.
- A *trusted* MIDlet suite can acquire some permission automatically depending on the security policy.

Permissions are used to protect APIs that are sensitive and require authorization. The MIDP 2.0 implementation has to check whether a MIDlet suite has acquired the necessary permission before invoking the API.

MIDlets do not acquire permissions explicitly through a code; rather, they acquire a protection domain. A protection domain is a set of permissions and interaction modes; those permissions can be either automatically granted (allowed permissions) or deferred (user permissions) until user approval.

Each protection domain, except for the untrusted domain, is associated to a set of root certificates. When signing a MIDlet suite, it is necessary to use a public key certificate that can be validated to one of those root certificates. This association will be used to assign the MIDlet suite to a given protection domain. The relationship between root certificates and protection domain involves that a domain can be associated to many root certificates, whereas a root certificate can be associated to only one domain.

The MIDP 2.0 specification has four protection domains for GSM/UTMS devices:

- *Manufacturer:* the manufacturer domain uses root certificates belonging to the device producer.
- *Operator:* the operator domain is used for the network operator MIDlets and may use root certificates available on storages (such as SIM cards).
- *Trusted third party:* the trusted third party domain will encompass well-known Certificate Authorities' (CA) root certificates.
- *Untrusted:* the mandatory untrusted domain does not have an associated root.

The MIDP 2.0 specification defines as untrusted a MIDlet suite for which the origin and integrity of the application cannot be verified by the device. This means that the access to restricted operations requires explicit user permission. If the device can verify the authenticity and integrity of the MIDlet suite and assign it to a protection domain, then the MIDlet suite is said to be trusted. A trusted MIDlet suite will have its requested permissions granted according to its protection domain [5].

IV. P2PBLUETOOTH

Our P2PBluetooth is essentially a mobile-to-mobile applications that enables file sharing among mobile phones. To implement our proof-of-concept solution, we have chosen to adopt a fully decentralized infrastructure with no third party or central server holding information about the system. Every mobile phone publishes its own list of available files and directly requests what desired by the user to mobile phones around by using Bluetooth connectivity.

Downloads are requested by specifying keywords; the adopted solution guarantees a complete autonomy and little user interaction. At this purpose, initially we thought of displaying the list of possible files made available by another device to the user, giving her/him the possibility to manually choose what she/he wants to request. However, this would have required continuous control and interaction with the device, contradicting the usability key requirement for this kind of software; for this reason, this feature was not implemented in the final version.

Once running, P2PBluetooth automatically initiates a search by querying other Bluetooth enabled devices in proximity for their list of available contents. Once an answer is received, the data of interest is automatically transferred. All this process is done without user mediation, the user only needs to specify the data identifiers needed for the download.

The macro-elements composing our system are represented in Fig. 1; they are also summarized in the following.

ConnectionManager is responsible for handling the connectivity of the system, both detecting new peers in proximity and determining when peers have become disconnected (through timeouts). Currently only Bluetooth connectivity is managed, however, other connectivity means could be added in the future.

P2PListener is an active service which listens for incoming client connections satisfying their needs; in essence, it is a server. The J2ME platform provides a native support by which a server component can externally publish information about itself; we have exploited this option to make data identifiers visible to other peers.

P2PPublisher manages the list of contents that the users intends to make available to other peers. The list is advertised when the system detects other peers around.

P2PDiscoveryService is an active daemon which periodically searches for contents made available by other devices in proximity, both hosting the P2PListener. The search, or inquiry, as it is referred by the Bluetooth specification, is part of the Service Discovery Protocol (SDP) protocol included in the Bluetooth protocol stack. The inquiry process is performed in the following sequential steps:

a) a device inquiry is started;

b) each detected device that hosts the P2PListener is queried, this phase is referenced to as service inquiry;

c) published contents are retrieved.

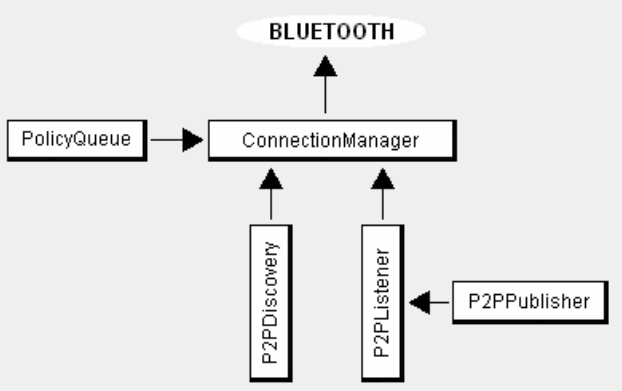

Figure 1. P2PBluetoot's macro-components.

PolicyQueue allows to control the number and the way requests are serviced. Two queues are managed: one for download requests and one for upload requests. Currently, each queue is locally processed in a FIFO fashion, whereas the global policy specifies that the two queues are processed in alternating order. Clearly, custom policies can be further added with a little effort.

A. *Content Search Functionality*

As we are dealing with a mobile phone and not a home/office computer, we have kept the content search strategy of P2PBluetooth as simple as possible; it can be summarized as follows:

a) all the lists of contents retrieved from the publishers are merged together;

b) the first element of the merged list that satisfies a download request issued by the user is then used for data retrieval;

c) point b) is repeated until all download requests issued by the user are satisfied or until there is no item satisfying that specific data request.

Figure 2. P2PBluetooth running on the Java Wireless Toolkit emulator.

As an example, Fig. 2 shows P2PBluetooth running on the Java Wireless Toolkit emulator. In particular, the user is selecting the "search" option from the menu, while, on the background, files currently available for download are visible. For the sake of conciseness, we have try to provide in this paper as many useful details as possible (given the limitation on the number of pages). However, the interested reader may find a video of a demo available online [6].

B. *Implications related to the Mobile Environment*

Further decisions taken during the analysis were influenced by the resource limitations of the execution environment provided in mobile devices. The Java community has released some best practices and how-tos as guidelines for efficient software design and coding [7].

In the considered scenario, the device's mobility can make the application freeze during transfer processing or queues grow. To deal with the former, the concept of timeout was introduced. Each download request issues a timeout; if an on-going download receives no response for a time longer than the timeout, the request is terminated and the involved device will be marked as disconnected; the download will be re-queued for later processing.

About the queue growth, initially, the system was built to simultaneously accept and service all requests. This proved not to be the correct approach: each processed request is handled in a separate execution thread and this can result in constant request processing and large memory utilization, that, due to the limited resources, cause the application to crash due to `OutOfMemoryException`. Calling the garbage collector in such a case is not the correct solution because the result of doing so is unpredictable. Hence, we had to limit the amount of memory being used at any time by queuing the requests and processing them one at a time.

C. *Security*

Given the security framework adopted by J2ME (see Section IV-A) and the type of permissions required by P2PBluetooth, the software needs to be executed within a trusted domain. For this reason, MIDlet signing is required.

However, in order to sign the MIDlet a code-signing certificate conforming to the X.509 Public-Key Infrastructure (PKI) is needed. In this first analysis of the subject a certificate was not acquired. Therefore, the development and testing of the product was performed on the emulator available in the Java Wireless Toolkit, which realistically emulates the environment and behavior of a mobile phone. It includes simple tools for creating a key pair and signing a MIDlet suite, so we were able to run and test the software in a trusted security domain.

An alternative solution to execute an unsigned MIDlet with the features of a signed one would be that of forging the operating system's specific component which handles the certificate validity check.

D. *The Current Release*

The mobile nature of the environment and the constrained execution environment are factors that need to be taken into consideration when designing the system

infrastructure. Also, the user's friendliness is important for the success of the application. Small devices fall into the category of consumer electronics, where user expectations are much higher.

Another important design issue is related to components such as routing and entity naming. Numerous P2P solution proposals can be found in scientific literature [8]-[10] that deals with these complex problems.

P2PBluetooth addresses some of the aforementioned problems, yet without aiming at providing an original solution. Indeed, the main contributions of our work are i) to propose mobile-to-mobile, proximity based, file sharing applications and ii) to demonstrate their feasibility through the creation of a proof-of-concept application.

We are well aware that to have it ready for the market, our P2PBluetooth would need further improvements and extensions. Yet, product serves well as a prototype application as it already presents important features, which are as follows.

1) *Simple, intuitive, and non-blocking GUI*
   - Ability to choose a keyword for your preferred download.
   - Ability to change the shared list of files.
2) *A configurable functional behavior*
   - User can change the maximum number of uploading connections.
   - The polling frequency is configurable.
   - Ability to serve multiple concurrent connections to the device.
   - Ability to limit upload/download bandwidth.
   - Add custom queuing policies with little effort.

Even if other communication means could be easily added in the future, yet we have decided to currently utilize just Bluetooth to create our proof-of-concept application. This choice was driven by the fact that, since its birth, Bluetooth has been intended for supporting communications among portable products (i.e., mobile phones) with limited battery power. This is confirmed by Fig. 3 that shows the energy consumption of Bluetooth versus Wi-Fi when employing two off-the-shelf communication chips: the CSR BlueCore2 and the Conexant CX53111, respectively.

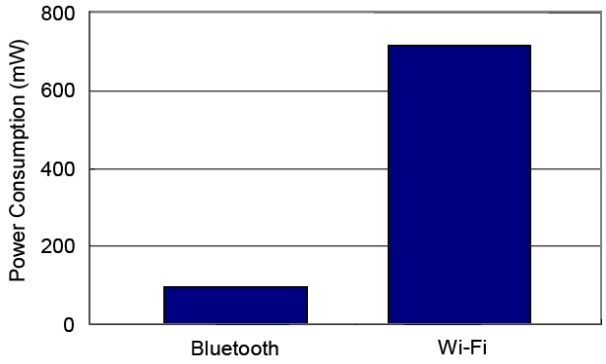

Figure 3. Power consumption comparison.

V. Conclusion and Future Work

The majority of people see Bluetooth as a cable replacement as well as a link between a handheld device and a hotspot that is connected to a local network or the Internet. Instead, we have exploited it to create a proximity-based file sharing application for mobile phones. We have chosen to use Bluetooth since its wide and free availability, its data rate, and its low energy consumption. Our application, named P2PBluetooth, has been created with J2ME to ensure the widest possible portability on mobile phones.

Through P2PBluetooth, we propose a new paradigm of use of the P2P file sharing service that merges this popular application with a ubiquitous tool, the mobile phone, to allow proximity-based file exchange. Exchanged data might be multimedia files for entertainment but also personal profiles or medical data to allow the creation of new proximity-based services that could improve our lives [11].

We are planning several future directions for this work. From a technical point of view, we intend to expand the possibility offered today by our P2PBluetooth in order to make it ready for a wide use in the real world. For instance, we need to add the possibility to interrupt and resume downloads, and to divide data in smaller chunks in order to download them from different users along the user's path.

From a utilization point of view, we plan to combine this work with other proximity-based services such as medical or social ones.

Finally, as said, a video of a P2PBluetooth demo, is available online at [6].

References

[1] T. G. Kanter, "Extensible Mobile Presence", in Proc. of the 4th International Workshop on Mobile and Wireless Communications Network, Stockholm, Sweden, Sep 2002.

[2] P. Tarasewich, "Designing Mobile Commerce Applications", Communications of the ACM, Vol. 46, No. 12, Dec 2003.

[3] B. Yang, H. Garcia-Molina, "Comparing Hybrid Peer-to-Peer Systems", in Proc. of the 27th International Conference on Very Large Data Bases (VLDB 2001), Roma, Italy, Sep 2001.

[4] Java API and Docs, http://java.sun.com/javame/reference

[5] Java Security Domains, http://wiki.forum.nokia.com/index.php/Java_Security_Domains

[6] P2PBluetooth, http://www.math.unipd.it/~cpalazzi/p2pbluetooth

[7] J2ME Best-Practices, Bluetooth communication protocol, http://www.nowires.org

[8] M. Caesar, M. Castro1, E. B. Nightingale, G. O'Shea, A. Rowstron, "Virtual Ring Routing: Network Routing Inspired by DHTs", in Proc. of SIGCOMM'06, Pisa, Italy, Sep 2006.

[9] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", in Proc. of SIGCOMM'01, San Diego, CA, USA, Aug 2001.

[10] T. Qiu and I. Nikolaidis, "On the Performance and Policies of Mobile Peer-to-Peer Network Protocols", in Proc. of CNSR'04, Fredericton, N. B., Canada, May 2004.

[11] S. Ferretti, S. Mirri, M. Roccetti, C. Sermenghi & V. Conforti, "Managing First Response Medical Aids With An Altruistic Web Application", in Proc. of the 3rd ICST/ACM/IEEE Pervasive Health 2009, London, UK, Apr 2009.