

# EMPIRICAL ASPECTS ON IMPLEMENTING APPLICATION SUPERNETWORKING

Jussi Ala-Kurikka, Mika Ylianttila, Erkki Harjula, Otso Kassinen  
MediaTeam Oulu Group, Department of Electrical Engineering,  
Erkki Koiso-Kanttilankatu 3, FIN-90014 University of Oulu, Finland

[jussi.ala-kurikka@ee.oulu.fi](mailto:jussi.ala-kurikka@ee.oulu.fi), [mika.ylianttila@ee.oulu.fi](mailto:mika.ylianttila@ee.oulu.fi), [erkki.harjula@ee.oulu.fi](mailto:erkki.harjula@ee.oulu.fi), [otso.kassinen@ee.oulu.fi](mailto:otso.kassinen@ee.oulu.fi)

## ABSTRACT

In this paper we provide a case study for implementing peer-to-peer (P2P) and application supernetworking into mobile devices. Paper portrays an empirical study of the implementation process with analysis and discussion. Evaluation of the implementation process is done in respect to system functionality and quality. Especially, implementation and communication quality of the current prototype implementations are evaluated. Case study illustrates various design choices in wireless and mobile application development. Application supernetworking and peer-to-peer communication concepts are discussed from the implementation oriented point of view. In this paper we focus on Java J2ME application programming experiences with JXTA and JXME technologies.

## I. INTRODUCTION

Development of wireless mobile applications has attracted a lot of attention during the recent years. There has been a major quest for “killer applications” that would boost the whole mobile industry and bring new revenues to the operators. An emerging hot topic is ability to share personal content over mobile Internet. P2P (Peer-to-peer) Software such as Napster, Gnutella, Kazaa and DC++ have implemented such functionality in PC environments for some time, but mobile variants providing this kind of functionality are still more or less missing. Also Instant Messaging and other popular P2P features in PCs have not received a breakthrough in mobile devices.

There are currently a number of implementation related restrictions in the current mobile devices and platforms, such as small screens, limited bandwidth, and relatively little storage space that have prevented from building large, complex page layouts [1]. However, as illustrated in Fig. 1, recent introductions of new mobile handset models are paving the way towards better resources for application development. Also, there are a number of software development kits (SDK) available for mobile application developers.

In particular, Java programming language with its edition to mobile devices, called J2ME, has attracted a lot of interest as an interoperable technology that interconnects various mobile phone models, PDAs and desktop computers supporting Java technology, thus being “pervasive” technology [2]. There has also been interest in implementing parts of Java in hardware which would provide greater run-time efficiency [8].

At the same time, mobile network and device specifications are becoming increasingly open. Some have stated that in the future there is no “mobile internet” but only one Internet where various end-user devices can access the same services. This paradigm shift results in that end-users can really experience the mobile networks and the open Internet becoming one. All borders in between the two are shaded in order to provide something new in 3<sup>rd</sup> and 4<sup>th</sup> generation mobile networks. This evolutionary process directs the services provided by mobile telephone operators to face a challenge presented by a new era: All-IP and Application Supernetworking. Application supernetworking in a P2P network is a disruptive technology that requires considerations on both technology and business. It provides new application scenarios to be used with mobile devices.

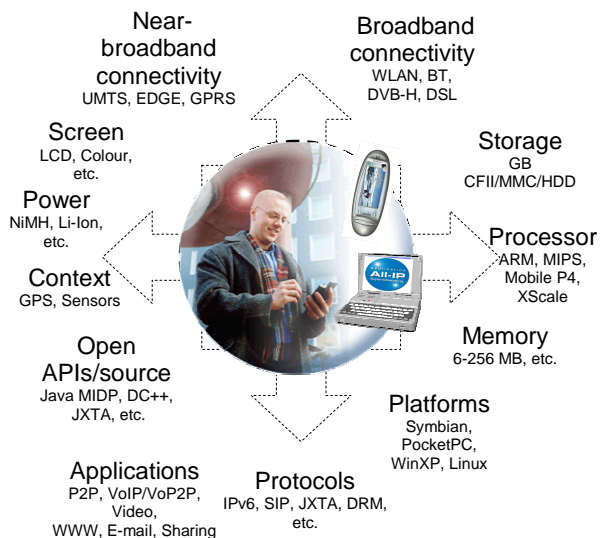


Figure 1. Enabling technologies for Application Supernetworking

In this paper the concept of Application Supernetworking is first discussed. Requirements for mobile P2P are discussed in section 2, and defined user and application scenarios for Application Supernetworking are described. Section 3 discusses the selections related to platform and test environment. Section 4 addresses empirical studies related to implementing mobile P2P. Evaluation of the implementation process is done in respect to system functionality and quality. Especially, implementation and communication quality of the current prototype implementations are analyzed. Discussion and analysis provides help and starting point for anyone starting developing mobile P2P applications. Conclusions are given at the end.

## II. REQUIREMENTS FOR IMPLEMENTING APPLICATION SUPERNETWORKING

Mobile P2P demands interoperability, platform independence and ubiquity among digital devices. Also security concerns must be taken into account in a totally new fashion. Inherent properties and requirements of new communication paradigm of Application Supernetworking can be defined as:

- All-IP
- New routing and service discovery (e.g., P2P)
- New ecosystem
- Device is a “communicator-computer” (GB mass memory, broadband connectivity, etc).
- Group management and semantic associations between persons and things
- Different domains of operation: “private space”, “enterprise”, “outdoors”, “group”
- Multi-domain, multi-device, multi-session

Application Supernetworking is a natural continuation of today’s mobile communication. Interaction today very rarely includes more than one media at once: calls have only voice, SMS messages only text. MMS messages are somewhere in the middle of 2G and 3G technologies. As we are moving towards using faster communication technologies and more capable end-user devices, the concept of application supernetworking can be fully implemented in the real world.

Application supernetworking contains at least three important elements [10]:

1. *Multisessions* and/or rich calls,
2. *Plug-and-play* interactions between sessions and applications
3. *Holistic connectivity* management

*Session* is an abstract entity which is created when one process starts communicating with another. A session can utilize various connectivity alternatives, such as WLAN or GPRS. Sessions can even be established between two processes running on the same device. Sessions are destroyed when communication is no longer needed. *Multisession* enables having multiple sessions open at any given time. The user might have e.g. two video conference call sessions, one voice only call, and a remote desktop session active simultaneously.

*Rich calls* enable multiple kinds of interaction over one phone call: you might for example want to have a video conference with audio, a file transfer and a shared desktop all running at the same time over one call session. A true supernetworking device can execute a number of processes using the same group of sessions in a multiplexed fashion, meaning application level session sharing and both multisession and rich call type of interaction, all simultaneously.

Application Supernetworking asks for easy-to-use and comprehensible uniform interfaces and standard ways of connecting applications in various kinds of ways. This

leads to considering *plug-and-play* interactions between sessions and applications.

*Holistic connectivity management* enables selecting the best connectivity available at a time (e.g., highest throughput, lowest end-to-end delay, lowest price). The visibility of the connectivity management to the end-user would be kept to a minimum by implementing the connectivity management as independent configurable middleware component.

As examples of Application Supernetworking, we have defined four user scenarios that are illustrated in the following.

### A. *Mobile file sharing*

File sharing between stationary and/or mobile devices with user group management and security control removes boundaries from the accessibility of data. Service providers could also make profit by selling temporary or permanent access to their chargeable shared data. Features such as remote download triggering (e.g. triggering a file download from someone else’s peer to the user’s home computer with a mobile phone) could prove handy in real-life situations.

### B. *Multi-device interaction*

Internet users may have several devices they use to access Internet and store content. Connection management and data synchronization among these devices is important already. We expand this scenario to consider P2P type of communication.

### C. *Real-time conference*

As mobile networks converge to an All-IP architecture transmitting voice over IP is clearly needed. This provides new possibilities for application supernetworking, such as conferences with simultaneous audio, video, text and application data.

Real-time conference can include simultaneous usage of chat or Instant Messaging [6] that let users write messages directly in real-time to each other. Examples of IM include Windows Messenger, Netmeeting and ICQ. There are already mobile clients available that support communications with these services, though they are not interoperable yet.

### D. *Process sharing*

When a user has started for example a real-time video conference session with a mobile device on the road, the session can be transferred without interruption to a more capable device in the user’s proximity, for example at home. In the desktop computer, Microsoft Netmeeting has provided desktop sharing functionality in the PC realm between different users. GoToMyPC, on the other hand, is an application for users who want to remotely access their PC’s desktop from anywhere. Users could be interested in these kinds of services in the mobile context as well. One scenario is a mobile user viewing another user’s document and commenting it by using pointing functionality and comment fields.

### III. EXPERIMENTAL SETUP

Our main target platform is Symbian OS based mobile phone. More specifically, we have selected Nokia Series 60 developer platform 2.0 as the first implementation platform. For example, Nokia 6600 phone meets these requirements. In addition to providing a possibility to install and run C++ programs inside SIS packages, the developer platform 2.0 supports Java 2 Micro Edition (CLDC) MIDP 1.0 and 2.0 [4,5] programs as well. Testing of J2ME applications can be done also with Compaq iPAQ devices installed with WLAN cards.

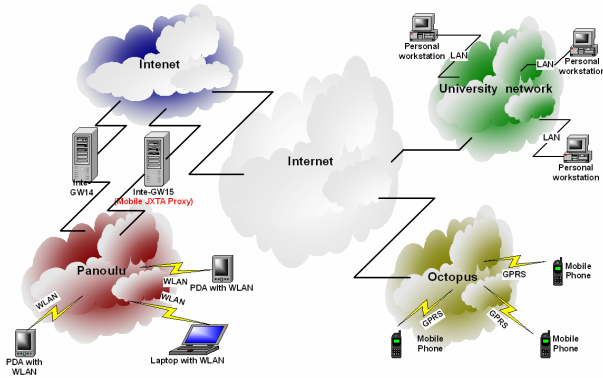


Figure 2. Topological Test Network Setup

Mobile devices are connected to an experimental network called Octopus which is part of the topological network setup illustrated in Fig. 2. Octopus provides for example EDGE (In Oulu area) and GPRS data services, SMS and MMS centers, WAP gateway, content delivery and streaming solutions, and presence server. WLAN access for iPAQs and laptops running both native client software and emulators is provided by panOULU, a city-wide wireless network in Oulu, Finland.

In the following we take a brief look at how the JXTA technology (including JXME) can match with various requirements for mobile P2P applications. JXTA technology is a set of open protocols that describe ways how different kinds of end-user devices can efficiently interact in a P2P fashion. JXTA communication is possible even with NATs and firewalls in between different nodes, or when the peers are using different network technologies. This would enable mobile P2P networking as such, but unfortunately JXTA (like many other P2P protocols) has not been designed with low hardware resources first in mind. Implementing fully to the JXTA set of protocols would prove to be too much of a burden for a mobile device, so some kind of a surrogate or proxy technology is needed.

JXTA protocols are text-based, so apart from the reference implementation written with Java there are also others available. One of these is JXTA for J2ME (JXME), which provides a lighter set of JXTA protocols for J2ME compatible devices. The so-called *proxied* JXME implementation relies on some other peer in the network to handle the most consuming responsibilities of a JXTA peer. The mobile implementation uses the services of a

JXTA relay and proxy to connect to the JXTA network of many kinds of devices. An example *proxied* application, JxtaMidpChat, is developed by the JXME community and is ready to be run on MIDP 1.0.

The *proxyless* version for MIDP 2.0 is not yet fully implemented. Proxyless requires more from the mobile device since it does not rely on a JXTA proxy. A prototype is ready for J2ME CDC (such as PDA devices). Among other JXTA implementations there is also a fresh JXTA-C implementation ready which is ported to e.g. Windows and Linux [6].

#### A. Messaging and routing

JXTA framework is divided in three layers: Core (includes mandatory low-level functionality), Services (not exactly required but highly desirable features such as file sharing) and Applications (high-level user applications). JXTA provides a set of protocols for P2P networks. These XML-based protocols include functions for e.g. peer discovery, endpoint routing, connection binding, basic query/response message exchange, and network propagation through rendezvous peers.

#### B. Peer group management

JXTA allows for management of peers individually and as a group. It allows for communication among peers via so-called pipes and also defines the message structure exchanged among peers in advertisements.

#### C. Security in P2P

In JXTA, security features such as authentication of peers and secure application data exchange can be implemented with TLS (formerly SSL) but P2P specific security issues like reputation-based trusted resource selection [7] must also be considered.

Our experimental setup consists of using the JXTA framework (with its various implementations) on desktop and laptop PCs, Nokia 6600 phones and Compaq iPAQs.

### IV. EMPIRICAL ASPECTS

This section describes empirical experiences about the wireless P2P and supernetworking implementation process.

#### A. Implementation process

After the definition of application user scenarios and the selection of implementation platform and programming tools, actual implementation process is started. It includes stages of software analysis and design, writing code, debugging and testing as a somewhat iterative process. Our implementation process is at design stage. We have kept a close eye on implementation development of the JXTA community. We have done tests on the different Java prototypes with PCs running either Windows (myJXTA and JxtaMidpChat for MIDP 2.0 on an emulator) or Linux (myJXTA), Nokia mobile phones 7650 and 6600 (JxtaMidpChat for MIDP 1.0 using Linux PC as a relay with proxy). Our current experience from

the testing on different platforms is discussed in the following section.

### *B. Comparing proxied and proxyless versions*

JXME proxied version uses a proxy and a relay that forward the messages coming from the JXTA network to the mobile device and vice versa. In addition to that, they filter out any unessential messages to save the mobile peer's network and processing resources. Proxyless JXME is about implementing a proxyless JXME client (in addition to others) for J2ME MIDP 2.0.

Combining small size and a rich set of features seems an impossible scenario for any one P2P platform, so one with many protocols and a possibility for relaying more constrained peer's messages with a more powerful one would look viable. After all, one cannot imagine having two separate P2P platforms – one for PCs and one for mobiles. This would essentially break the scene of a borderless, ubiquitous Internet where information would be accessible anywhere, anytime and with any device. The original idea behind JXTA is to unify P2P computing in a way HTTP did for the web [3].

Therefore proxied JXME might seem promising. It does only what is necessary in the mobile end, leaving the dirty work to more powerful devices (meaning relays and proxies). However, in the absence of such a service, or more likely in the absence of knowledge of where in the network topology such a service might exist, communication is not possible. This encourages the development of proxyless technologies for even more powerful mobile phones. A mobile phone of later model might act as a super peer (connection point) e.g. for a small Bluetooth JXTA community – no external relay or proxy would be needed.

The power consumption is, of course, always something to think of when designing mobile applications. We assume that the battery life is related to the amount of traffic. The proxied version has a configurable poll interval, at which intervals the JXME client contacts the relay to ask for new messages for itself. The default is one second. A more frequent polling interval would intuitively mean longer battery lifetime, but on the other hand the delay of course would increase as well. The proxyless version does not require such polling, thus less control traffic would be needed, resulting extended battery lifetime.

### *C. Implementation quality evaluation*

One must note that at the moment the JXTA programs are not of commercial quality. For example, we have bad experiences on leaving a JXTA relay running overnight over an X Window System connection on top of an SSH connection to a remote Linux machine. On several occasions the software has somehow collapsed and tried to start itself over, resulting in creation of dozens of orphaned Java processes on the remote computer. The user friendliness of the JxtaMidpChat application could also be better.

We had some trouble setting up the myJXTA application. The documentation for it is hidden very well, so JXTA project mailing lists have proven very handy. Both the configuration on the relay computer and on client PCs had too many undocumented settings before getting hands-on advice from the community. Failure to set up the relay correctly led in not too descriptive error messages in the JXME clients trying to connect to the relay. Also chat messages in the myJXTA application seemed to disappear sometimes if the configuration was not done properly.

It is also worth noting that the current JXME API is not very flexible. There are functions ready that provide for the needs of the example JxtaMidpChat and JxtaTicTacToe applications but not much more. You could also blame the J2ME MIDP sandbox model. The J2ME CLDC/MIDP sandbox model prevents a MIDP application from accessing the persistent storage of the platform. A Record Management System (RMS) is provided to enable MIDP applications to store their internal data, but access to files saved by other applications on the device is made impossible by using only MIDP. This holds true for the other way around as well: a native application cannot (without great effort) access the information saved by a MIDP application.

Some mobile phones manufacturers have provided their own, supplementary APIs that provide this functionality, but they are very rare, and they do not include Nokia. This, of course, restricts the possibilities of implementing P2P applications with MIDP so much that one has to really weigh what things are of value. For example, a file sharing application would be of very little use implemented using only the MIDP API. It is interesting to note, however, that this is due to security reasons while SIS packaged applications implemented with Symbian compatible C++ have access to virtually all the functionalities of the platform without restrictions. This substantial difference is perhaps not very well known by a typical end-user.

JXTA-C might prove the best choice in the end. A JXTA-C to Symbian C++ port would certainly provide the true flexibility of the JXTA framework for a mobile application. In fact, there is a community subproject doing the port, but unfortunately progress has been very slow. SIP will most probably also be included in our Symbian C++ implementation later on, when adding audio/video features.

### *D. Communication quality evaluation*

JXTA offers three types of pipes that transfer messages: unicast (point-to-point), secure unicast and one-to-many (propagate). Benchmarks at [9] reveal that with JXTA 2.0 and J2SE 1.4.1 the message round trip times (time from sending a message to another peer to receiving an ACK message) through different pipes are about 100 ms-200 ms with typical message sizes (1 KB-100 KB). Relays in between the communicating peers add approximately 50ms-100 ms extra each. Normal JXTA messages are XML formatted so XML parsing causes its part of the overhead. Although not measured, logically the roundtrip times could have been even an order of magnitude shorter

in the case of non-P2P, direct TCP or UDP communication. Therefore applications suffering the most from communication delays (e.g., remote desktop control) present a challenge to P2P platforms such as JXTA.

Proxied JXME handles client-side firewalls (that let outbound messages through but block some inbound messages) with no problems because of its polling nature, but in our configuration a firewall was also present on the relay server side. The relay in panOULU network did not receive messages coming from the JXME device connected to the Octopus GPRS network. This naturally blocked connections between the peers.

There is a project called VoP2P within the JXTA community standing for Voice over P2P, thus indicating that there is some overlap between JXTA and SIP. One might expect that SIP and JXTA perform best in their original tasks, SIP handling multimedia sessions and JXTA resource sharing. A more complete comparison of SIP and JXTA is given at [10].

However, there is a work being done to expand and cross boundaries. If a powerful future mobile device would know both SIP and JXTA and also how to transfer files with SIP and how to establish video conferences with JXTA, it would certainly provide a more flexible solution and would be compatible with perhaps significantly more peers (knowing possibly only one of the two protocols). There might also be other protocols the device should be capable of communicating with. This may require another protocol to handle application protocol selection in a plug-and-play manner. Tackling this dilemma with a broad spectrum of possible protocols and making more implementation experimentations remains as future work.

## V. CONCLUSIONS

In this paper we have provided a case study for implementing peer-to-peer (P2P) and application supernetworking applications into mobile devices. Evaluation of the implementation process was done in respect to system functionality and quality. Especially, implementation and communication quality of the current prototype implementations were evaluated. Implementation process included definition of application user scenarios, followed by selection of implementation platform and programming tools, and actual software development.

Case study illustrated various design choices in wireless and mobile application development with JXTA technology. Application supernetworking and peer-to-peer communication concepts were discussed from the implementation oriented point of view. Java J2ME application programming experiences were discussed and analyzed. Study showed that a new, currently unavailable JXTA C++ implementation for Symbian OS would be a highly relevant choice for implementation platform in mobile application supernetworking. J2ME MIDP and JXME, on the other hand, proved to be too restricted for true supernetworking applications. Further work includes

more implementation related studies on application supernetworking, peer-to-peer architectures, networking related aspects with vertical roaming scenarios, and implementation of mentioned user scenarios with various protocols and further scenarios.

## ACKNOWLEDGMENTS

The acknowledgments are due to the project team of Application Supernetworking/All-IP project. The authors would like to thank TEKES, Nokia, Elektrobit, TeliaSonera, Serv-IT and IBM for supporting this project financially.

## REFERENCES

- [1] K. Read, F. Maurer, "Developing mobile wireless applications", IEEE Internet Computing, Vol. 7, Issue: 1, Jan.-Feb. 2003, pp. 81 – 86.
- [2] S. Helal, "Pervasive Java, Part II", IEEE Pervasive Computing, Vol. 1, Issue: 2, April-June 2002, pp. 85 – 89.
- [3] W. Yeager, J. Williams, "Secure peer-to-peer networking: the JXTA example" IT Professional, Vol. 4, Issue: 2, March-April 2002, pp. 53 - 57
- [4] D. Kochnev, A Terekhov, "Surviving Java for mobiles", IEEE Pervasive Computing, Vol. 2, Issue: 2, April-June 2003, pp. 90 – 95.
- [5] C. Enrique Ortiz, Eric Giguère, "Mobile Information Device Profile for Java™ 2 Micro Edition", Wiley Computer Publishing, 2001.
- [6] D. Bradbury, "Messages in an instant", Application Development Advisor, November-December 2001, pp. 18-20.
- [7] A. A. Selçuk, E. Uzun, M. R. Pariente, "A Reputation-Based Trust Management System for P2P Networks", International Workshop on Global and Peer-to-Peer Computing, 2004, p. 1.
- [8] D. Hardin, "Real-time objects on the bare metal: an efficient hardware realization of the Java™ Virtual Machine", In the Proceedings of the Fourth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 2001. ISORC - 2001., 2-4 May 2001, pp. 53 – 59.
- [9] E. Halepovic, R. Deters, "JXTA Performance Study", IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM03), Canada, 2003.
- [10] D. Howie, et. al, "State-of-the-art SIP for Mobile Application Supernetworking", in the proceedings of NRS/FWCW conference 2004,Oulu, Finland, August, 2004.