Extending Friend-to-Friend Computing to Mobile Environments

Sven Kirsimäe, Ulrich Norbisrath, Georg Singer, Satish Narayana Srirama, Artjom Lind
Institute of Computer Science, University of Tartu
J. Liivi 2, Tartu, Estonia
Sven.Kirsimae@ut.ee, Ulrich.Norbisrath@ut.ee, Georg.Singer@ut.ee, Satish.Srirama@ut.ee, Artjom.Lind@ut.ee

***Abstract*—Friend-to-Friend (F2F) computing is a popular peer to peer computing framework, bootstrapped by instant messaging. Friend-to-Friend (F2F) Computing is a simple distributed computing concept where participants are each others friends, allowing computational tasks to be shared with each other as easily as friendship. The widespread availability of applications and services on mobile phones is one of the major recent developments of the current software industry. Due to the also emerging market for cloud computing services we mainly find centralized structures. Friend-to-Friend computing and other Peer-to-Peer (P2P) computing solutions provide a decentralized alternative. However these are not very common in mobile environments. This paper investigates how to extend the Friend-to-Friend computing framework to mobile environments. We describe our process and point out possible pitfalls and achievements. For this investigation, we ported our private cloud environment Friend-to-Friend Computing to Android and Symbian. We demonstrate a mobile gaming application using Friend-to-Friend Mobile.**

***Keywords*—Distributed Computing; Peer-to-Peer; Social Networks; Android; Symbian; Mobile Software; Mobile Networking and Management**

I. INTRODUCTION

In this paper, we will present a case study on how we ported our F2F Computing framework to Android and the Symbian S60 mobile platform. Based on our observations we will outline the process of making P2P computing frameworks mobile aware.

Along with the increased availability of software as a service (SaaS) on PCs, services are also moving into the mobile market. Moreover, a smart phone nowadays has become a commodity device with millions subscriptions worldwide. It is not just a mere voice only device anymore, but offers many alternative communication technologies. The importance of running applications and accessing services becomes the key demand from the users. Also having access to these in a convenient manner plays an important role. Users do not care about installation process, installation locations–on the phone or in the cloud–, only about the provided functionality. They appreciate the possibility of having a set of self selected applications on their devices. The importance of this application and service support becomes evident with the success of the iPhone [1] and Android platforms as well as with the struggling of Nokia [2] to provide a competing architecture. In addition, Nokia's investment in Maemo and Meego, opensourcing and withdrawing Symbian and now the switch to Windows 7 at the same time as well as Google's success with Android prove the same point. Their strategy is to offer other ways to distribute applications and services as an alternative to Apple's very popular and successful but proprietary platform.

The approach of providing computational resources from smart phones for various collaborative tasks is conceptually similar to providing services on them. This was studied at the mobile web service provisioning project [3], where Mobile Hosts were developed, that provide basic services from smart phones. Mobile Hosts enable seamless integration of user-specific services to the enterprise by following web service standards, also on the radio link and via resource constrained smart phones [4].

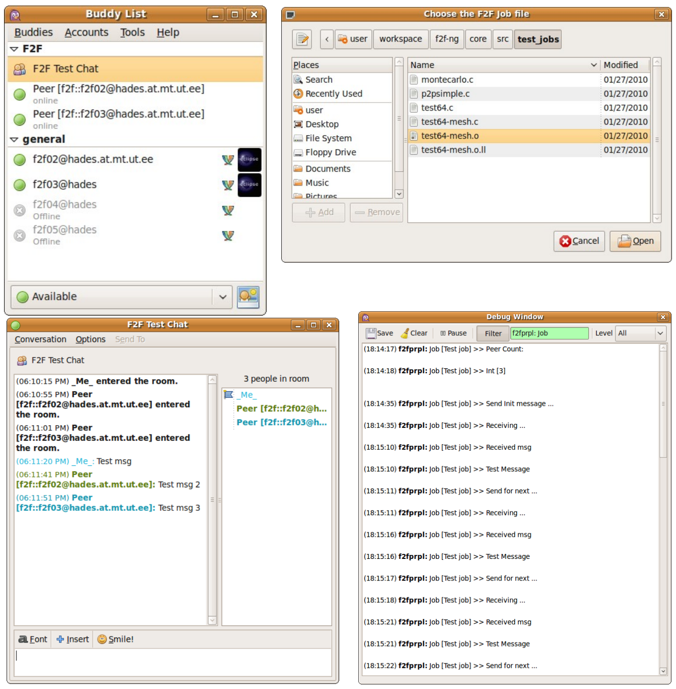
Friend-to-Friend (F2F) Computing is a simple distributed computing concept where participants are friends or acquaintances of each other, allowing computational tasks to be shared with each other as easily as friendship. It will be explained in more details in Section II. A network of friends is the base for F2F Computing. A similar concept can be seen among mobile device users. People using these devices are often socially connected. There are multiple services for mobiles available nowadays, which support this concept. Examples for such services are Facebook, Twitter, Flickr, blogging, youtubing, or multiplayer games.

In this paper we describe the simply installable extension of F2F Computing called F2F Mobile running on the Android and Symbian mobile operating systems. We will show the achieved transparency between mobile and static systems from a developer's point of view. We will show some of the difficulties in extending P2P computing to the mobile environment, which have to be addressed by other mobile platform developers. We will also present an application demonstrated on top of our F2F Mobile.

The rest of the paper is organized as follows. Section 2 will give a brief outline of Friend-to-Friend (F2F) Computing. Section 3 introduces the F2F Mobile concept, its implementation details, the demonstrated application, and a small reference of criteria for mobile platform selection and implementation issues. Section 4 shows related work for F2F and mobile environments. Section 5 concludes the paper with future research directions.

II. FRIEND-TO-FRIEND COMPUTING

Friend-to-Friend (F2F) Computing was initially motivated by the complexity of setup, usage, and administration of Grid



```
F2F
  F2F Test Chat
  Peer [f2f::f2f02@hades.at.mt.ut.ee]
  online
  Peer [f2f::f2f03@hades.at.mt.ut.ee]
  online
general
  f2f02@hades.at.mt.ut.ee
  f2f03@hades
  f2f04@hades
  Offline
  f2f05@hades
  Offline
```

Name	Modified
montecarlo.c	01/27/2010
p2psimple.c	01/27/2010
test64.c	01/27/2010
test64-mesh.c	01/27/2010
test64-mesh.o	01/27/2010
test64-mesh.o.ll	01/27/2010

```
F2F Test Chat
(06:10:15 PM) _Me_ entered the room.
(06:10:55 PM) Peer [f2f::f2f02@hades.at.mt.ut.ee] entered the room.
(06:11:01 PM) Peer [f2f::f2f03@hades.at.mt.ut.ee] entered the room.
(06:11:20 PM) _Me_: Test msg
(06:11:41 PM) Peer [f2f::f2f02@hades.at.mt.ut.ee]: Test msg 2
(06:11:51 PM) Peer [f2f::f2f03@hades.at.mt.ut.ee]: Test msg 3

_Me_
Peer [f2f::f2f02@h...
Peer [f2f::f2f03@h...
```

```
(18:14:17) f2fprpl: Job [Test job] >> Peer Count:
(18:14:18) f2fprpl: Job [Test job] >> Int [3]
(18:14:35) f2fprpl: Job [Test job] >> Send Init message ...
(18:14:35) f2fprpl: Job [Test job] >> Receiving ...
(18:15:10) f2fprpl: Job [Test job] >> Received msg
(18:15:10) f2fprpl: Job [Test job] >> Test Message
(18:15:11) f2fprpl: Job [Test job] >> Send for next ...
(18:15:11) f2fprpl: Job [Test job] >> Receiving ...
(18:15:16) f2fprpl: Job [Test job] >> Received msg
(18:15:16) f2fprpl: Job [Test job] >> Test Message
(18:15:17) f2fprpl: Job [Test job] >> Send for next ...
(18:15:18) f2fprpl: Job [Test job] >> Receiving ...
(18:15:21) f2fprpl: Job [Test job] >> Received msg
(18:15:21) f2fprpl: Job [Test job] >> Test Message
(18:15:22) f2fprpl: Job [Test job] >> Send for next ...
```

Figure 1. F2F Pidgin plugin screenshots: F2F Group friends, F2F Chat, application selection and debug window

networks. Encouraged by the fact that Skype made voiceoverIP (VoIP) usable for everyone and these deficiencies of current Grids, we combined the best parts of both ideas into a system today known as Friend-to-Friend Computing. F2F Computing is an open source project for spontaneously running distributed applications using the resources provided by computers of friends. The initial setup of the virtual parallel or distributed computer environment makes use of Instant Messaging (IM) software as the triggering means. As a result, people can share computation tasks and other resources in an easy and intuitive manner through their social connections.

Creating an F2F Computing network consists of specifying a set of IM contacts and starting distributed applications or services. An important aspect of F2F Computing is its ability to be plugged into various different instant messengers and its independence of and interoperability between different instant messaging protocols. F2F Computing tries to choose from a set of network protocols the fastest one to connect each peer with each others. It also uses Network Address Translator (NAT) traversal techniques to accomplish efficient connections in case of address translating network devices. Figure 1 shows a collection of screenshots of dialogs of the F2F Pidgin plugin. It shows the list of friends added to a F2F Group, an F2F Chat, the application selection, and a debug window of a running F2F application.

The new architecture of F2F Computing (see Figure 2) allows different clients for setting up F2F networks and running applications or services on them (F2F Adapters). Most commonly used are plugins for IM (we have plugins for Pidgin and SIP Communicator – now Jitsi [5]). To run an F2F network the plugins need to be installed on the devices of all participants. One of the group members has to initiate the F2F Group by adding initial participants. After authorization by these, applications or services can be started as jobs. There are also command-line clients for starting initiators without user interaction.

Once the F2F network is established, the clients can exchange files between their friends in their contact lists, use collaboration tools like a whiteboard, play games, or accelerate computational intensive tasks like rendering or research simulations. Computational applications that have been carried out on F2F Computing include Monte Carlo computations, distributed matrix multiplication making it possible to solve large distributed systems of linear equations, and rendering tasks on Blender [6].

The first version of F2F Computing [7] was written in Java and was realized as a plugin of the multi protocol instant messenger SIP Communicator (Jitsi). F2F Computing was later rewritten to have a lower footprint. The core is now implemented in C allowing it to be ported even to restricted platforms like mobile devices. On a standard x86 system, the core itself has now only a size of 34k. We also re-implemented the application layer. This means, we implemented the instant messaging adapter as a plugin for Pidgin and as a Python command line client. For the execution adapter, which executes the tasks, we initially used Python. Python is here only used as a prototype for the access to the actual core. Currently, we are adapting Low Level Virtual Machine (LLVM) [8] and Java Virtual Machine (JVM) as execution adapter to allow other languages (like C, C++, and FORTRAN) for the executed applications and services. Python is only used here to access the F2F API.

The F2F Computing framework needs fast communication between participating peers (friends). The research was triggered by the need for fastest available communication for distributed desktop computing (cycle scavenging) applications. However, the fact that also other IM applications like VoIP, Video over IP and file transfer, are in real need for direct and fast communication availability, F2F Computing has grown into other application domains as well. We have successfully used F2F Computing for the aforementioned computational applications, for teaching, a cross-IM-brand whiteboard application, two computer games and a file transfer application.

Figure 2 shows the F2F Computing architecture. The framework consists of four layers: F2F enabled applications, application, adapter, core, and communication. In the adapter layer we provide the abstraction from the different communication providers of the communication layer in one uniform interface for F2F Computing applications, which can be run via the Computing adapter as byte-code compiled from various languages. This abstraction provides access to the concepts service, peer, and group.

One of the important requirements of F2F is reliable connectivity between peers and speed of communication. Therefore the framework has various connectivity possibilities like the TCP communication provider, reliable UDP communication provider, or the IM communication provider. In the future, the communication layer can be easily extended by implementing the specific communication providers (for example, *Bluetooth*, *Infiniband*). The framework chooses always the fastest way. If

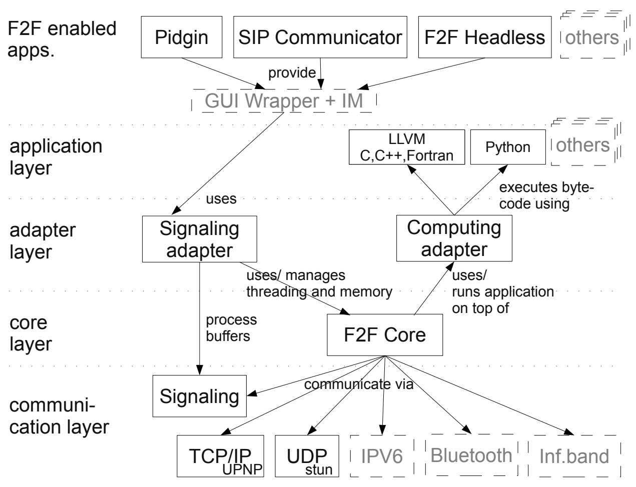


Figure 2. The F2F Computing architecture

peers are in the same local network direct TCP should be possible. If it is not, the framework attempts to establish UDP connection using NAT traversal techniques implemented in the UDP communication provider. In the worst case, if peer-to-peer connectivity is impossible, the messages are sent through instant messages of the respective instant messenger.

III. F2F Mobile

F2F Computing is based on a simple and transparent social concept: being friends or acquainted with others. It also offers an easy way to share resources on demand between a network of such friends. The mobile phone is one of the most important appliances for supporting social interaction. The demand for services and applications ubiquitously supporting this social interaction is growing [9]. However, due to the strongly fragmented nature of mobile technology, current service and application development is complex and focused on single services or products. Our hypothesis is that F2F Mobile will introduce a transparent environment for social applications and services. F2F Computing allows a spontaneous creation of social networks and the deployment of applications and services within this network. Including mobile devices in such a network is an obvious step.

Consider the use case where a group of people want to share some content in private and independent of third party services. Having F2F Computing embedded into mobile devices allows them to spontaneously create a private network of friends in order to share content or run an application. F2F Computing relies on a third party service only while bootstrapping the P2P network. After direct connections were established the communication is happening only between the peers in the private group.

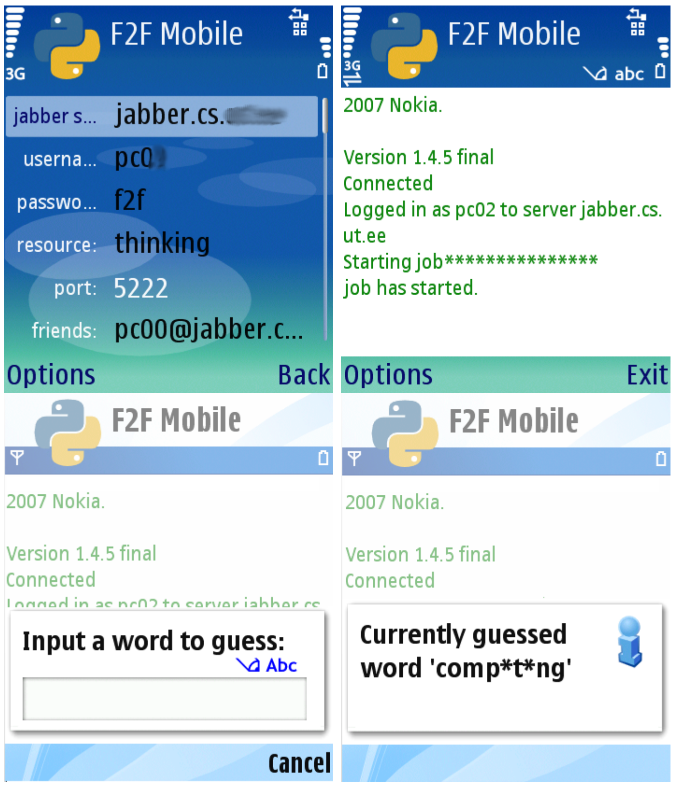
Before designing F2F Mobile, we considered two major factors – the market penetration of mobile platforms on the one hand and the simplicity of third party development of mobile platforms on the other hand. At the time of carrying out this research, Nokia's Symbian, Google's Android, and Apple's iPhone OS had approximately the same share of applications on smart phones. However in short time Android and iPhone OS have significantly gained importance and currently Nokia is far below 50 percents market share. Today we would suggest to target the Android platform first as it comes with less restrictions especially in a legal sense than Apple's iPhone OS. It is not allowed to run a virtual machine, like for example Python, without violating the EULA. As Android was not that popular and not that feature-rich (there was no possibility to run native code) at the time we started our research, we selected Nokia's Symbian S60 for our F2F Mobile prototype. It provided a proved C development environment, Python support, and accessibility to the devices. However, the next F2F Mobile port will be Android based, the development process will be addressed later in this section.

F2F Computing is currently mainly based on C and several language interfaces. Symbian S60 supports by default a development in C and C++. A Software Development Kit (SDK) [10] and a development environment based on eclipse [11] are provided for free. A well maintained Python port [12] exists. However, the build of Symbian binaries is supported well only on Windows platforms, and this was a biggest disadvantage, as our research lab is Linux-based. In comparison Android SDK is supported on all three platforms (Windows, Linux and Mac OS).

Mobile devices are usually very restricted concerning their hard- and software. Applications are event-driven rather than multithreaded. In Symbian multithreading is possible and is used inside the Operating System, but it is generally avoided in applications, because it potentially creates several kilobytes of overhead per thread. Therefore, we avoided threads in the Symbian port. Because of memory limitations, applications are restricted in comparison to standard PC systems in their memory allocation strategy. The heap might be only several Megabytes and the stack be even smaller. Allocating memory dynamically can easily lead to termination of the application, system crashes, or kernel panics. Therefore, as a first step, the core of F2F Computing was rewritten completely in ANSI C without threading and with static memory management. Symbian does not provide the standard C libraries by default. We used Open C/C++ [11] to provide the missing C libraries. For the Python adapter and Python execution environment, we had to add more missing C libraries and the corresponding Python interfaces. To allow simple installation of the F2F Mobile the imported libraries were packaged additionally into the F2F Mobile package, also including the F2F Core, and the Python F2F Adapter as application layer (Figure 2).

Another limitation is that the S60 SDK emulator only emulates but not replicates the mobile device. In some cases it even represents fewer constraints (memory, access rights) compared with the real mobile device. Developing, testing and debugging with the SDK means that the application needs to be re-tested on the real devices in the real environment to make sure it is behaving as expected.

For the installation, there have to be four packages installed (in this order): Python for S60 3rd Edition 1.4.5, corresponding Python shell, Open C/C++, and the F2F Mobile package. The actual receiving client Python script comes as a separate file. It will be started from the shell. After logging in to



jabber s... jabber.cs.[redacted]
userna... pc0[redacted]
passwo... f2f
resource: thinking
port: 5222
friends: pc00@jabber.c...

2007 Nokia.

Version 1.4.5 final
Connected
Logged in as pc02 to server jabber.cs.ut.ee
Starting job***************
job has started.

2007 Nokia.

Version 1.4.5 final
Connected
Logged in as pc02 to server jabber.cs

Input a word to guess:

2007 Nokia.

Version 1.4.5 final
Connected

Currently guessed word 'comp*t*ng'

Figure 3. Screenshots of demonstration application on F2F Mobile.

F2F Mobile core, it will wait for the application–the F2F task– to arrive. The application itself can be sent either from another mobile device (with a server script) or from an instant messenger with the F2F plugin installed on a full PC. The application arrives to the mobile client and executes on all the arriving peers and automatically makes the spontaneous F2F network infrastructure available for all participants. As one sample application we implemented a small hangman style game taking approximately 200 lines of code. The application runs both on mobile and PC platform. At the moment the GUI is still coded separately for PC and mobile platform. In this game one player thinks of a word and all other players have to guess it in turns, either guessing a new letter or the whole word. If somebody guesses the word, this player can think of the next word. Figure 3 shows some screenshots of the start up and running the game on two different mobile devices. The first two screenshots were taken on an Nokia E70, the second two in the S60 Emulator. The setup we used here were a PC with the emulator, an E70, an E61, and a PC with Pidgin and the F2F Plugin. We submitted the application from the PC. The first two screenshots show the configuration and application deployment phase, the last two screenshots are taken while the game is running.

Table I and Table II summarize our experiences, when porting F2F Computing to Symbian S60. Table I summarizes possible problems porting a platform to a mobile environment. Table II shall be a reference for the technical realization of such a port. It depicts our problems and selections we had to make to achieve the port.

We did some experiments with Maemo and could prove that we can compile the F2F Computing with Python adapter directly. This is not surprising as Maemo is basically a far less stripped down Linux than Android.

criterion	comment
platforms to consider	Symbian, iPhone, Android, Blackberry RIM, Windows Mobile, Maemo [13], Meego
market penetration, for personal target market	• Take into account current penetration and make platform decision accordingly • Development/change of the penetration over time (growing, shrinking?)
maturity of platform	How established is the platform? Has there been done already a lot of development on it? Is there an active community of developers? iPhone OS and Android are for example still young players, but have a very big and fast growing community.
distribution channels	How can applications be uploaded to a phone, is it only possible via an application store, do other possibilities exist? Is the kind of material, which can be distributed, restricted?
non functional requirements	• Are used programming languages supported? • Does an SDK exist? Is it free or not? • Is a development environment provided? Is it free or not?
legal issues	• License fees, are all interfaces available, which are needed? • Consider legal technical restrictions (for example no virtual machine allowed on iPhone) • Application signing issues, can only signed applications be run?, how difficult/expensive is signing? • Do you use open source libraries (GPLv3), that you will not be allowed to use in a potential restricted environment? • Are there usable open source (OS) components? These can avoid a lot of problems and simplify development very much.

Table I
SELECTION CRITERIA FOR A TARGET MOBILE PLATFORM.

Addressing the criteria, which are summarized in the tables, we present our initial experience with the F2F Computing Android port. Portability and performance are only two of the many advantages of Friend-to-Friend Computing. Its modular architecture allows to build different sets of installations in order to fit specific platforms. In the Android port we reuse the F2F Core component with only minor changes to the code. These changes are mainly in regard to providing the Java interfaces. The graphical user interface (front-end) can then be implemented in Java using Android specific widgets. Having a platform specific front-end helps utilizing the corresponding platform related features like touch screen, drag and drop, scaling, or 3D rendering. Utilizing these features is essential to provide user-friendly GUI.

The performance is granted by the core component written in C. During implementation of the Symbian S60 port, we faced multiple issues based on limits of S60 platform (Open

criterion	comment
threading	Try to avoid threading, some platforms might have problems.
multi-tasking	There might be different levels of multi-tasking available. iPhone: no multi tasking for applications, Maemo full pre-emptive.
memory management	Low footprint, avoid dynamic memory allocation.
test on real device	The Symbian emulator has different restrictions.
languages	Choose a good balance between languages you use, try to have an abstracting core or library, try to use a wrapper.
library dependencies	Check the dependencies and try if referenced libraries are either available or compilable. We had problems with hashlib and Pyexpat (XML).
datatype sizes	Make sure to wrap datatypes, so they can be used in a mobile environment. Usually the size has to be fixed in number of used bits.
ensure simplicity of installation	See, how the packaging mechanism for you target platform works. Is it possible to pack all in one package?

Table II
CRITERIA TO ACCOUNT FOR IN THE IMPLEMENTATION PHASE.

C/C++ library, small memory and avoiding the threads usage). According to Android documentation there is a Bionic library [14] (custom libc) and Native Development Kit (NDK) [15] starting from Android 1.5. While taking a closer look at the Bionic library we found incomplete support of the POSIX threads and C++ exceptions. The pthread_cancel() is not supported and pthread_once() is limited as we use not these functions neither C++ in F2F Core component implementation it will be well portable to Android platform featured with Bionic library. We add the Android tool-chain to our build scripts. This is not an issue due the Python based SConstructor (SCons) [16] builder that we use. There is an exhausting manual how to provide a new tool-chain for SCons [17], there is also project [18] using combination of SCons and Android NDK. Considering legal issues, there is no special licenses needed for Android development and the development tools are free to download for Linux, Windows and Mac.

Android uses Java classes (Widgets) to provide an API for writing user interfaces. In order to wire the Java based user interface and the native C-code of F2F Core and the corresponding C routines are exposed to Java using Java Native Interface (JNI). The entire process can be completely automated with Simplified Interface Wrapper Generator (SWIG) [19]. The same solution was applied to expose F2F Core routines to the Python back- and front-end in the F2F mobile prototype for Symbian S60.

The computing engines are essential to remote-execute the code in F2F. For Symbian there was Python support, for Android there are two ways: either we build standard Python from sources [20] using NDK or we use the scripting-layer (SL4A) [21]. SL4A allows to access Android Native API using various scripting languages including Python. In addition it is possible to run scripts from native code using Intent Builders of the Android SDK. This means that we are able to run the remote Python Scripts on Android platform. Having the Python support is essential to have compatibility with existing Symbian S60 port and the Desktop builds of F2F Computing. As Android is Java based there is not much effort needed to provide Java computing engine for F2F Android port. The F2F Core component is loaded into JVM (Dalvik) at runtime, corresponding Java routines are accessible using standard JNI methods (FindClass, GetMethodID, CallVoidMethod, CallIntMethod).

Another project we carried out on Android was porting LLVM with the purpose to execute the native code remotely on Android platform. The LLVM port to F2F was successful on the Desktop platforms, therefore the next step to keep compatibility is to introduce it on the mobile platforms. As LLVM supports multiple languages (C/C++/Fortran/Java/Python) porting it once to Android allows us not care about running remote Python or Java code.

Android standard and native development kits are supported by major platforms (Windows, Linux and Mac OS X). In addition there is an Android development plugin for Eclipse.

These experiments and observations with Android show that the modular architecture of F2F Computing is well suited to be ported to Android.

IV. RELATED WORK

There exist several groups trying to realize concepts similar to F2F. We took a look at the akogrimo [22], [23] project. It claims to achieve the step “from Cluster Grids toward Mobile Collaborative Business Grids”. It outlines a similar vision to our F2F Mobile on several whitepapers. Case studies, market analysis, and real deployed networks are missing. Therefore, the actual differences in software and implications from the mobile fragmentation are not addressed.

F2F Computing focuses mainly on the mobile client side and on the option to spontaneously setup private cloud environments. Conceptually, it does not distinguish the type of device that is participating in the F2F Computing network. Therefore, F2F Mobile works similar to the mobile clients for public clouds, yet helps in sharing resources and CPU cycles of individual mobiles. Thus F2F Mobile looks similar to P2P Computing, but we still distinguish the general perception of P2P and F2F Computing. For example, Boinc [24] is regarded as a P2P Computing application. It allows harvesting the CPU cycles via a central mediator. Even if it is called a P2P Computing solution, it only facilitates P2P in a collaborative sense but still uses a star topology and therefore becomes an example for a client server architecture. In that sense F2F is more related to P2P than these traditional applications that are thought to be P2P.

Regarding moving P2P based systems to the mobiles we have studied other projects, ex: LightPeers [25]. LightPeers provides a good proposal of the lightweight P2P platform with a well defined architecture. They also propose P2P protocols that are minimalistic and easy to implement. They studied other existing P2P protocol libraries (JXTA, JXME, Proem) and identified the problems with adapting these technologies for the mobile environment, thus were proposing their own set of architecture and protocols. However, there is not much information provided about the implementation as well as the way to distribute and parallelize applications that run on top of the LightPeers platform.

V. Conclusion and Future Work

With this paper, we showed how the F2F Computing platform can be extended to mobile devices. This drift supports establishing mobile private clouds with significantly less effort. The paper listed several tricks and difficulties and choices taken to avoid them in building F2F Mobile. The paper also showed some applications that were implemented, proving the technical feasibility of the concept.

Not only proving the concept of F2F Mobile, the study also provides several guidelines in building such systems. Especially with the tables provided in Section 3, we highlighted the points in selecting the destination mobile platforms and architectural choices to be taken care of in building application and services. This study will generally be useful for any community that is working and building private clouds and distributed computing platforms that have in foresight shifting their platforms to mobiles.

While the first prototype of F2F Mobile is ready, it provides a lot of scope for further research. Our efforts are especially directed to the possibility of writing code in any language and be able to run it on mobiles. For this, we are developing support of other execution adapters like LLVM [8] in addition to Python. This will allow us to support multiple languages for development and in terms of hardware architecture heterogeneous execution environments. A further important goal is to create a release candidate for Android. If Apple changes some of its software restrictions an iPhone port would also be possible. Furthermore, we are also interested in building more applications and services for F2F Computing, especially in collaborative, mobile gaming, and m-learning domains. Another huge issue is a standardization of some GUI elements offered transparently in the F2F environment. Due to the severely different screens and use patterns on the client devices such an abstraction will be a challenging research area.

Because of the strong fragmentation in terms of hardware, software, and operators, there still remain many problems with transparent on demand software deployment across multiple static and mobile participants in today's networks.

VI. Acknowledgement

This paper was supported by the European Social Fund through the Estonian Doctoral School in Information and Communication Technology.

References

[1] 148Apps.biz | apple iTunes app store metrics, statistics and numbers for iPhone apps. Available from: http://148apps.biz/app-store-metrics/ [cited July 19, 2011].

[2] Nokia's application store faces apple dominance. *Time*, May 2009. Available from: http://www.time.com/time/business/article/0,8599,1900901,00.html [cited July 19, 2011].

[3] S. N Srirama, M. Jarke, and W. Prinz. Mobile web service provisioning. In *Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services*, page 120, 2006.

[4] S. N Srirama and M. Jarke. Mobile hosts in enterprise service integration. *International Journal of Web Engineering and Technology*, 5(2):187–213, 2009.

[5] Jitsi (sip communicator). Available from: http://www.jitsi.org/ [cited July 19, 2011].

[6] Blender. Available from: http://www.blender.org/ [cited July 19, 2011].

[7] U.Norbisrath, K.Kraaner, E.Vainikko, and O.Batrashev. Friend-to-Friend computing - instant messaging based spontaneous desktop grid. In *Internet and Web Applications and Services, International Conference on*, volume 0, pages 245–256, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[8] C. Lattner and V. Adve. LLVM: a compilation framework for lifelong program analysis & transformation. In *Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization*, 2004.

[9] J.Steele. comScore: mobile internet becoming a daily activity for many. Available from: http://www.comscore.com/Press_Events/Press_Releases/2009/3/Daily_Mobile_Internet_Usage_Grows [cited July 19, 2011].

[10] R.Edwards and L.Barker. *Developing Series 60 Applications: A Guide for Symbian OS C++ Developers*. Pearson Higher Education, 2004. Available from: http://portal.acm.org/citation.cfm?id=983988.

[11] Nokia developer - qt development frameworks. Available from: http://www.developer.nokia.com/Develop/Qt/Qt_technology.xhtml [cited July 19, 2011].

[12] J.Laurila, M.Marchetti, and E.martt. Python for s60. Available from: https://garage.maemo.org/frs/?group_id=854 [cited July 19, 2011].

[13] Home of the maemo community. Available from: http://maemo.org/ [cited July 19, 2011].

[14] Bionic c library overview. Available from: http://www.netmite.com/android/mydroid/1.5/bionic/libc/docs/OVERVIEW.TXT [cited July 19, 2011].

[15] Android NDK android developers. Available from: http://developer.android.com/sdk/ndk/index.html [cited July 19, 2011].

[16] SCons: a software construction tool. Available from: http://www.scons.org/ [cited July 19, 2011].

[17] SCons new target platform and new toolchain. Available from: http://buildman.net/sbuild_man/porting_eng/new_platform.htm#Step_2 [cited July 19, 2011].

[18] AllJoyn - proximity-based peer-to-peer technology. Available from: https://www.alljoyn.org/ [cited July 19, 2011].

[19] Simplified wrapper and interface generator. Available from: http://www.swig.org/ [cited July 19, 2011].

[20] python-for-android - Py4A - google project hosting. Available from: http://code.google.com/p/python-for-android/ [cited July 19, 2011].

[21] android-scripting. Available from: http://code.google.com/p/android-scripting/ [cited August 11, 2010].

[22] S. Wesner, T. Dimitrakos, K. Jeffrey, and H. Performance. Akogrimo-the grid goes mobile. *ERCIM news no59 2004*.

[23] C. Loos. E-health with mobile grids: The akogrimo heart monitoring and emergency scenario. *EU Akogrimo project Whitepaper*, 2006.

[24] D.P. Anderson. BOINC: A system for public-resource computing and storage. In *proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, pages 4–10. IEEE Computer Society, 2004.

[25] B.Guldbjerg Christensen. Lightpeers: A lightweight mobile p2p platform. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pages 132 –136, 2007.