Comparison of Linux and Symbian Based Implementations of Mobile Peer-to-Peer Environment

Arto Hämäläinen, Petri Hiirsalmi, Jari Porras

Lappeenranta University of Technology

P.O. Box 20

53851 Lappeenranta, Finland

{Arto.Hamalainen, Petri.Hiirsalmi, Jari.Porras}@lut.fi

Abstract: In this paper two software implementations of a mobile peer-to-peer communication environment (PeerHood) are compared. The basic properties of the environment are first identified and then the implementations are presented. First one of the implementations is based on Linux operating system and is implemented on HP/Compaq iPAQ devices. This implementation is compared to the PeerHood implementation on Symbian OS based mobile phones. Due to limitations of the Symbian environment and devices some properties has been left out in mobile phone environment, but it still has certain similarities with the Linux implementation.

1. INTRODUCTION

Mobile communications has increased its popularity for the last 20 years. In the early days the development happened in the area of communication technologies. NMT, GSM, GPRS are all results of the need for mobility. Currently we are in a situation where we have several mobile communication technologies available. Bluetooth, WLAN and GPRS are all supported by the modern mobile devices. If considering the usage of devices there are not too many differences between mobile phones and PDAs. It seems that they are evolving into a single personal trusted device (PTD) that the user uses for different purposes. What is more important than the actual device at the moment is how the users are using those devices. Mobile phone manufacturer Nokia announced their Linux based networking device few months ago. This device and simultaneously announced development platform, Maemo [1], reflect how not only the devices and their usage but also the development for mobile devices is evolving. Mobile devices has traditionally been proprietary systems. Recent years have brought some changes. A significant step in opening up the mobile device development has been the introduction of Symbian OS in smart phones, and open source platform takes the idea further. Linux is a strong candidate for future mobile phones, if certain shortcomings and requirements for mobile phone use can be satisfied [2].

The rapid development and wide utilization of the fixed Internet based services has led into a situation where Internet access is the key factor in mobile devices as well. Fixed Internet has seen development in personal communications in form of instant messaging, presence and peer-to-peer communications. All these will eventually find their way to mobile Internet as well. Our vision is that the next step will be the true peer-to-peer type of communication in mobile environment. This could be referred as ubiquitous or pervasive computing / communication where devices are communicating directly with each other. This is very logical step as short-range technologies like Bluetooth have been implemented on almost all new mobile devices. Therefore in this paper we are studying the peer-to-peer type of communication in mobile environment. We have implemented PeerHood, a mobile peer-to-peer environment, for Linux based PDA devices and Symbian OS based mobile phones. In this paper we want to compare the software issues of these implementations.

2. PEERHOOD ENVIRONMENT

PeerHood is an implementation of a personal area network (PAN) based on mobile peer-to-peer paradigm. There exist several approaches for the personal networking ranging from proximity based approaches, e.g. PeerHood [3][4] and Digital Aura [5], to approaches based on personal interests, e.g. Personal Networking [6], I-centric communications [7] and Personal Distributed Environment [8]. Our approach is based on the need of local services (proximity) and the use of them through different networking technologies (Bluetooth, WLAN and GPRS).

The goal of the PeerHood system is to provide a communication environment where devices act and communicate in a peer-to-peer manner. This means that devices communicate directly with each other without any centralized servers. In order to enable fast creation of required ad-hoc type networks the immediate neighbors of a device are monitored and the gathered information is stored for possible future usage. The second goal is to create a library that enables usage of any supported networking technology via a unified interface so that the underlying networking structure is hidden from the applications point of view. As a direct consequence the application development time should be reduced because complex tasks like device discovery, connection establishment and error checking are handled by the PeerHood system.

In our approach the Personal Trusted Device (PTD) is continuously sensing its neighborhood through different network technologies and maintains the gathered information for the further usage. As other devices are noticed they are added into the neighborhood of the PTD. Figure 1 presents a view to the layers of our implementation. The seamless connectivity or the Networking layer handles the basic operations of the PeerHood i.e. it searches, monitors and maintains information of the changing pervasive environment. As new devices and services are observed they are stored into a neighborhood information table. Thus the PTD always has up-to-date information concerning its environment. Information concerning the environment is used and also partly managed by the middleware elements. For example, the personal information element provides information to the services for personalization purposes but also sets the limits for the gathered information. There is no need to gather information of services not interested by the user.
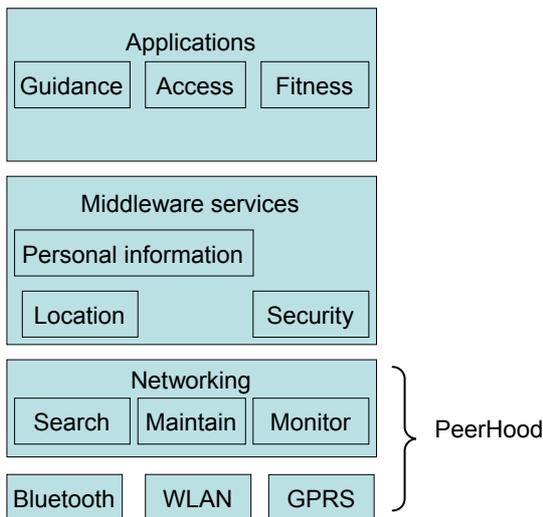

Figure 1. Layers of Personal Trusted Device model, including PeerHood networking layer.

Before implementing the mobile peer-to-peer ideas into mobile devices, the key functions of the PeerHood system were defined. The following list presents these functions:

- *Device discovery* – PeerHood must be able to detect other PeerHood-capable devices that are within the range and belong to the same neighbourhood. How the device detection is done, depends on the underlying networking technology and is left as an implementation specific issue.
- *Service discovery* – PeerHood must be able to detect what services are available on a remote device and what attributes they contain. How the

service information is transferred between the devices is implementation specific as long as the proper packet structure and message flow are followed.

- *Service sharing* – PeerHood has to offer a mechanism for applications or additional middleware components to use and register services. PeerHood advertises the available services to other devices using networking technology dependent mechanisms.
- *Connection establishment* – PeerHood must offer a way to connect two or more devices in the same PeerHood neighbourhood. The connection procedure must be transparent so that from upper layer's point of view the underlying networking technology doesn't affect the procedure. The only exception is that technologies that on the hardware level allow only one connection at a time are not required to support multiple simultaneous connections. PeerHood should offer a streaming socket interface to handle the established connection providing *MAbstractConnection* interface.
- *Data transmission between devices* – PeerHood must support data transmission between connected devices via objects that implement the *MAbstactConnection interface.* The programmer is in charge to ensure that endianess and word length issues are handled in a proper way. In other words, the PeerHood should not care about the content of the transferred data.
- *Active monitoring of a device* – It must be possible to put some device in the PeerHood neighbourhood under active monitoring. This means that a requesting application or middleware component is notified as soon as the monitored device goes out or comes to the range. Proper response time and range are technology dependent issues.
- *Seamless Connectivity* – Seamless Connectivity defines a technology for changing the active networking technology automatically if the established connection weakens or breaks. Seamless Connectivity tries to always provide the best possible connection for the user while maintaining connectivity.

3. LINUX AND SYMBIAN AS DEVELOPMENT ENVIRONMENTS

One of the most crucial choices for a software developer is the selection of the device and operating system to work with. The number of target devices and possible users is economically a significant factor, but technically the developer's familiarity with the target device and most importantly the suitability of the device for desired function

play great roles. Mobile Linux devices, iPAQ handheld devices with Familiar Linux operating system, were selected as the initial PeerHood environment mostly because of earlier experience in the Linux operating system and its BlueZ Bluetooth protocol stack. Bluetooth stack is available both in i386 and ARM processor families, and changing the application development environment is quite straightforward. It's important to follow guidelines [9] when selecting tools to use in development to ensure flawless configuration, compilation, installation and verification phases of cross compilation. Cross compilation still requires a bit of manual work, but tools like Scratchbox [10] are of great help when trying to automate the process. Scratchbox provides sandbox type of approach to develop for several target devices. Applications for different devices can be developed simultaneously without harmful interaction of environments. Development for Symbian phones is a bit more complex process. Linux application could be run natively on the build PC or automatically on target device through Scratchbox, whereas Symbian program had to be run in emulator or transferred to a mobile phone for testing.

3.1 Linux Implementation of the PeerHood

The main component of Linux implementation of PeerHood is the Peerhood daemon, PHD. PHD is the component, which takes care of all the functions of PeerHood, such as device and service discovery and connection establishment. The PHD uses different network technologies through PeerHood plug-in interface. Currently, three plug-ins have been developed for Linux PeerHood: Bluetooth, WLAN, GPRS. PeerHood can be used by the applications via PeerHood library interface. Communication between the library in application process and the daemon process is established with local sockets through socket interface. System software components are illustrated in Figure 2.
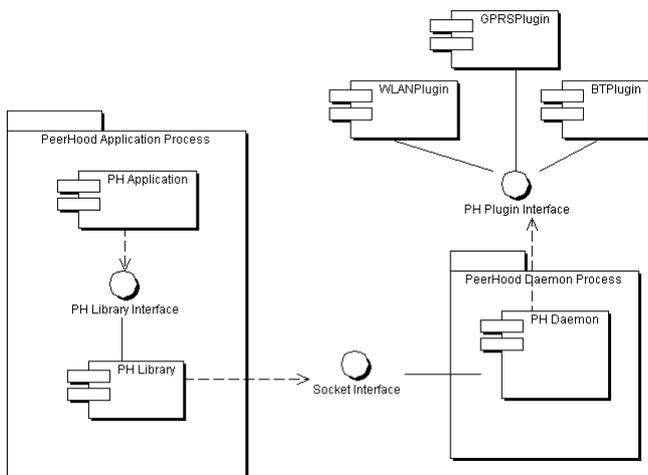

Figure 2 Components of PeerHood for Linux

Because the daemon and the library are independent components, PeerHood class structure is also divided into two parts. Important part of whole PeerHood structure is the abstraction between networking specific classes and interfaces. PeerHood allows developers to add new plug-ins which can be used by the daemon. To be able to provide this architecture, the whole technology-specific functionality must be contained in one class which implements specific interface. This allows the core components to remain the same while adding new functionality using plug-ins. Daemon class diagram is illustrated in Figure 3.

CDaemon is the main class which uses different plug-ins and connection objects. *CDaemon* uses the network specific plug-ins through *MAbstractPlugin* interface to search other mobile devices and services and to advertise own device and provided services. *CDeviceStorage* acts as a repository for found devices and services. *CDeviceStorage* follows the Singleton design pattern which makes sure that there is always only one instance of it present. Daemon uses network specific connection objects through *MAbstractConnection* interface. Connection objects are needed by plug-ins to exchange service and device information between devices.
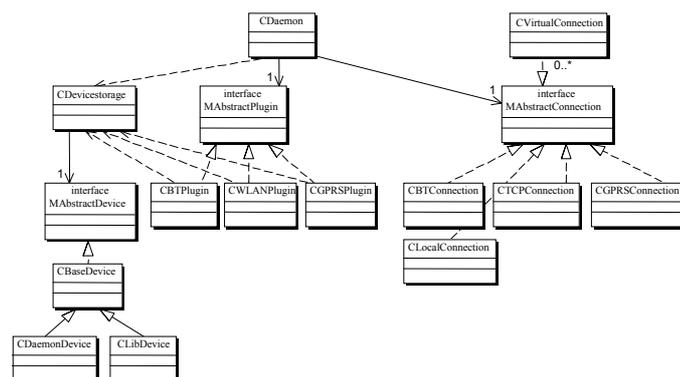
Figure 3 Daemon class diagram of Linux implementation

The PeerHood library (Figure 4) handles general PeerHood operations, most of which are internal. Instances of connection objects are created using the *Factory* class and existing connections are handled by the *Engine* class. Library also performs miscellaneous functions, e.g. logging and monitoring. *MpeerHood* interface provides third-party applications a unified interface to the PeerHood library. The interface is actually Implemented in *CPeerHoodImpl* class. This class provides methods for callback creation and registeration, application-daemon connections and eventually connections to remote devices.

Figure 4 Library class diagram of Linux implementation

3.2 Symbian Implementation of the PeerHood

From the beginning it was decided that the Symbian OS version should not be as complete as the Linux version. This implementation was developed to test if peer-to-peer solutions can be used on mobile phones at all. Most fundamental difference from the Linux version is that public Symbian SDK doesn't (officially) allow creation of background daemon applications. This means that the daemon-based solution used in the Linux version is impossible to create in Symbian OS. Because of this limitation, the whole PeerHood subsystem, with the exception of plug-ins, has been implemented as one module (Figure 5). Currently, only plug-in available in Symbian environment is Bluetooth plug-in.

Figure 5 Components of PeerHood for Symbian OS

Figure 6 shows the class diagram of PeerHood's Symbian OS version. This structure has certain similarities with the Linux version but also some fundamental differences. Just like in the Linux version there is one main interface, *MPeerHood*, that third-party applications are developed against. This pure virtual interface is implemented in the class *CPeerHoodImpl* that is the main class of this PeerHood version. It is responsible for connection handling and active object scheduling. Active objects provide a simple way to handle asynchronicity in Symbian OS.

Figure 6 Class diagram of Symbian OS implementation

3.3 Comparison of Linux and Symbian Implementations

The basic approach for a application developer is similar, whether he wishes to use either Linux or Symbian implementation of PeerHood. First an instance of running PeerHood implementation class must be fetched using *GetInstance()* function of *MpeerHood* class. Then desired functions can be called through this PeerHood library interface. Basic functions provided by the PeerHood library can be seen in Table 1.

Table 1 – Basic functions provided by the PeerHood library

Init	Initializes the PeerHood object
GetDeviceListL	Returns list of remote devices
GetLocalServiceListL	Returns list of local services
RegisterService	Registers a service
UnregisterService	Unregisters a service
Listen	Listens a port for connections
Connect	Connects to a remote device or service
Disconnect	Closes a connection

Figure 7 shows a PeerHood application example. The application initializes the PeerHood environment, fetches the list of devices and iterates through the list printing out the device names.

```
#include <Peerhood.h>
#include <iostream>
#include <cstdio>

int main(int argc, char** argv) {

  MPeerHood* peerHood = MPeerHood::GetInstance();
  if (!peerHood->Init(argc, argv)) {
    std::cerr << "PeerHood initialization failed"
              << std::endl;
    return EXIT_FAILURE;
  }
  TDeviceList* list = peerHood->GetDeviceListL();
  for (TDeviceIterator i = list->Begin();
       i = list->End();++i)
  {
    std::cout << Found device "
              << (*i)->GetName() << std::endl;
  }
  return EXIT_SUCCESS;
}
```

Figure 7 PeerHood application example

Basically the main difference between the implementations is, that Linux implementation includes a background daemon process, Peerhood daemon (PHD), whereas PeerHood-enabled Symbian OS application loads the PeerHood components itself when the library interface is loaded by the application. In Linux environment, PHD accepts local connections from applications and enters their services into the service database. Multiple applications, served by one PHD, can run in one computer and listen different ports. In Symbian OS implementation, only one PeerHood enabled application can be running at a time. Therefore the design changes from the local device point of view from client-server (applications-PHD) to all-in-one. Programming-wise, slight differences exist between function and variable names in Linux and Symbian implementations, mostly because different naming conventions. Application developer, who is familiar with either one of the implementations should not run into trouble while introducing himself to the other environment.

4. CONCLUSION

In this paper we have presented two implementations of a mobile peer-to-peer communications environment (PeerHood). The first one is based on Linux and is implemented on Compaq iPAQ PDAs. The second implementation is based on Symbian OS and is evaluated on Nokia mobile phones featuring Series 60 Platform. A set of operations defined for the PeerHood environment has been implemented in both implementations. Due to limitations of Symbian OS environment and devices some differences still exist in implementations. Furthermore, Symbian OS implementation still lacks plug-ins for WLAN and GPRS, as well as few other properties which are available in Linux implementation. From the application developer's point of view, two implementations still resemble each other closely and changeover between them should not lead into problems, if operating systems themselves are familiar to developer.

REFERENCES

[1] Nokia: Internet Tablet 2005 Software Edition and the Maemo Development Platform, Introductory White Paper, 2005. Available at: http://sw.nokia.com/id/5bcf36fa-819f-44b5-a2ba-3c9d7409b993/Maemo_Developer_WP_v1.0_en.pdf .
[2] Nakamoto, Y.: Toward Mobile Phone Linux, Proceedings of the 2004 conference on Asia South Pacific design automation: electronic design and solution fair, 2004.
[3] Porras J., Hiirsalmi P. and Valtaoja Ari: Peer-to-peer communication approach for mobile environment, 37th IEEE Annual Hawaii International Conference on System Sciences (HICSS), 2004.
[4] Porras J., Jäppinen P., Hiirsalmi P., Hämäläinen A., Saalasti S., Koponen R. and Keski-Jaskari S.: Personal Trusted Device in Personal Communications, 1st International Symposium on Wireless Communication Systems, 2004.
[5] Ferscha A., Hechinger M., Mayrhofer R., dos Santos Rocha M., Franz M. and Oberhauser R.: Digital Aura. Available at: http://www.soft.uni-linz.ac.at/ Research/Publications/_Documents/digitaleAura.pdf .
[6] Niemegeers I. and Hememstra de Groot S.: Personal Distributed Environments for Mobile Users, Mobile Europe 2003, European conference on mobile solutions, 2003. Available at: http://katanga.bbn.de/mobile_europe_2003/ .
[7] Popescu-Zeletin R., Steglich S. and Arbanowski S.: Pervasive Communication A Human-centered Service Architechture, Proceedings of the 10th IEEE International Workshop on Future Trends of Distributeed Computing Systems (FTDSC'04), 2004.
[8] My personal Adaptive NET (MAGNET), Available at http://www.ist-magnet.org/ .
[9] Robb, S.: Creating Cross-Compile Friendly Software, Proceedings of the Linux Symposium 2004, Volume Two, 2004, pp. 449-460.
[10] Mankinen, V. and Donselaar, V.: Scratchbox – Making embedded Linux cross-compilation faster and easier, Technical whitepaper, Available at: http://www.movial.fi/client-data/file/ 20040330_Movial_Scratchbox_white_paper.pdf .