Peer-to-peer Communication Approach for a Mobile Environment

Jari Porras, Petri Hiirsalmi and Ari Valtaoja
Lappeenranta University of Technology
P.O. Box 20
53851 Lappeenranta
Finland
{Jari.Porras, Petri.Hiirsalmi, Ari.Valtaoja@lut.fi}

In this paper an approach to the peer-to-peer communication in a mobile environment is presented. This approach is based on the concept of personal trusted device that constantly monitors its neighborhood and collects information for the use of different applications. The reference implementation is built on top of iPaq PDAs running the Linux operating system. Class diagrams are presented for the main elements of the environment and the properties of the prototype are presented. Although this approach is not limited to any particular communication technology only Bluetooth technology is considered and presented in this paper.

1 Introduction

The world of digital communications is changing rapidly. Not only new technologies evolve, but new communication paradigms and new ways of combining the existing solutions are created. These developments will increasingly affect the users as they will use new services through some new device using some new communication technology. In the following subsections some of these changes are discussed.

For the last decade the mobile phones and personal digital assistant (PDA) devices have satisfied different needs. Mobile phones have been used mainly for the communication purposes whereas PDA devices have contained applications merely capable of information storage and modification. The recent developments in technology have brought these devices closer to each other. As new services evolve the properties traditionally available in PDA devices need to be implemented into the mobile phones. This will eventually lead to a situation where only one so called Personal Trusted Device (PTD) is sufficient. In this paper both mobile phones and PDAs are referred as personal trusted device that is used for all types of services.

New services are not only a reason for the development of terminals but they may also be consequences of the development. As terminals evolve they will enable new service scenarios and new business models [1]. Positioning and personalization are seen as examples of the current trends. In the future the content of a service will depend increasingly on the location of the terminal as well as the preferences of the user. Concepts like body are network (BAN), personal area network (PAN) and personal network (PN) are likely to be implemented in the near future [2].

The implementation of the BAN, PAN and PN approaches is strongly related to the development of different communication methods and technologies. Bluetooth, Wireless Local Area Network (WLAN), General Packet Radio Service (GPRS) and Universal Mobile Telecommunications Service (UMTS) have all their own roles in the future of communication. Due to the limited range, Bluetooth technology seems to be suitable for local communication purposes [3].

As the locality becomes more important and the technologies like Bluetooth evolve, the use of peer-to-peer approach will become more popular. Currently peer-to-peer approach is limited by the operation of the fixed network but this will change when this approach is applied into mobile networks. Peer-to-peer communication is seen as one of the most promising communication paradigms of the future [4, 5] as it allows resource sharing in a flexible manner. It allows truly dynamic networking [6, 7] and offers possibilities for the service scenarios of the future.
This paper presents an approach that combines the current trends, namely a personal trusted device centered personal communication environment that is based on peer-to-peer communication paradigm. Although this approach is not limited to some specific communication technology only Bluetooth technology is considered and presented in this paper. A reference implementation is built on top of iPaq PDAs that are used as personal trusted devices.

2 Peer-to-peer Communication Concept on a Mobile Environment

For peer-to-peer communication in fixed network there exist several approaches. Three different approaches, i.e. Napster model, power server model and CPU power sharing model, are presented in [8]. Unfortunately these approaches do not take advantage of the mobile environment. There also exist some approaches to the mobile peer-to-peer communication [1, 9, 10, 11 and 12]. Unfortunately they all look the problem from a different angle than our approach. [1] concentrates on distribution of information over the mobile peer-to-peer network, [9] studies authentication, authorization and accounting (AAA) problems, [10] looks the problem with a much larger scope and [12] has a different goal. Closest to our proposal comes [11] where an approach of virtual device is presented. The Virtual device idea itself can be considered to work on top of our environment.

Our approach is based on a hierarchical structure where the peer-to-peer communication paradigm is build on top of wireless networks. Our approach concentrates on a very local peer-to-peer communications. Thus PAN type of a network approach is used. The core of our approach is a personal trusted device, e.g. mobile phone or PDA, which is considered as an independent element in the mobile environment. The personal trusted device is continuously sensing its neighbors in the changing mobile environment. This changing environment works as a personal area network for the personal trusted device and it is called as peer-to-peer neighborhood, or PeerHood. This idea is presented in Figure 1 as the cloud of devices around the personal trusted device.

PeerHood may use different network technologies but the implementation varies according to the properties of the selected technology. For example in a Bluetooth environment the neighborhood contains only devices that are sensed by the personal trusted device whereas in a WLAN environment the concept of neighborhood contains elements within the area of the same access point. This would implement the idea of Personal Network as presented in [2]. PeerHood enables roaming between different network technologies assuming that the same service is available through them.

In order to take advantage of the PeerHood some middleware modules, e.g. personalization module, has been defined on top of the PeerHood. These modules offer some value adding services for the applications implemented on the personal trusted device.

2.1 General Implementation of the PeerHood

Figure 2 presents the internal structure of a generic PeerHood implementation. Actual implementations may vary depending on the target platform’s characteristics but the principles are the same anyway. The main objective of the PeerHood is to know the immediate neighborhood of the personal trusted device. As the personal trusted device may be connected by using different network technologies, the PeerHood needs to consider the special properties of all those technologies. In order to support multiple...
network technologies the PeerHood architecture is pluggable. This means that all network related functionality is isolated into plugins, one for every network technology. However, in this paper only Bluetooth technology is considered due to its simplicity for peer-to-peer communication.

---

**Figure 2. PeerHood implementation.**

As the PeerHood monitors different network interfaces it collects information concerning the neighbors. This information may contain elements on the areas of the devices and their services. These two types of information are divided into different tables.

- Information of the actual devices is stored into the *Neighborhood info* table according to the inquiry or search procedure. This information is updated by using separate monitor or ping function. The neighborhood information table should contain the updated information of the actual neighborhood all the time.

- Service information is stored into the *service info* table. This information is gathered from the devices in the neighborhood by using service description function, e.g. Service Discovery Protocol (SDP) in Bluetooth. Other approaches may also be considered.

PeerHood offers an interface to the middleware modules or applications on top of this environment. This interface offers a set of services to the upper layers. Applications can utilize the gathered information by querying the PeerHood tables for some service they are interested. In addition they can use PeerHood to receive real-time information about their present connections.

Figure 3 presents an example of the general behavior of the PeerHood concept. In this example personal trusted device with PeerHood client *PH_Client_1* is monitoring its neighborhood and it detects another device *PH_Client_2*. This new device is asked if it supports the PeerHood concept. If so, then this device is added into the *Neighborhood info* table and the available services are queried. The answer is saved into the *Service info* table. In this example a Hello World service is found and this service is used.

---

**Figure 3. Operation example of the PeerHood concept.**

### 3 Linux Implementation

The first implementation of the PeerHood system was developed for Compaq iPaqs running the Familiar Linux distribution. This combination of hardware and software was chosen because of its great flexibility and ease of programming using openly available tools. Figure 4 presents the four main components of a fully functional Linux implementation and also the relationships between them. The main components are: PeerHood Daemon, PeerHood Library, Additional Components and finally PeerHood Application. Of these components the daemon and the library are mandatory for any application to work. Additional components are optional middleware libraries that further extend PeerHood’s functionality. No additional components were developed during the prototype implementation process.
The most fundamental part of the whole PeerHood system is the daemon. It has several important responsibilities including the discovery of other devices, advertisement of services and resources available on user’s own device and monitoring the status of device’s neighborhood. These functionalities were implemented as a daemon because several applications might very well use the PeerHood system simultaneously. As such, there’s no mean to duplicate heavy and resource consuming operations like device discovery for each application instance. In addition, the daemon-based approach enables applications to receive more up-to-date neighborhood information because the information is available immediately when an application is launched.

When the Bluetooth plugin is started, it inserts the PeerHood tag into the device’s SDP database. Other Bluetooth-enabled PeerHood devices use this tag to recognize the presence of PeerHood on the device. If the plugin detects PeerHood on some other device it sends a query to that device. The other device’s PeerHood daemon then replies with a list that contains all the services and resources available on that device. When the response is received the plugin inserts the collected data to the CDeviceStorage. After that, local applications can use the retrieved data.

If a new PeerHood-capable device is located the plugin will monitor its presence so that there’s always accurate information about the nearby devices. In the prototype plugin the monitoring is implemented so that the device discovery procedure is ran on periodic intervals. If a device is not found during three sequential discoveries it is considered to be out of range and its information is removed from the CDeviceStorage.

Data transmission between devices is done by using the Logical Link and Adaptation Protocol (L2CAP), which is located right above the Host Controller Interface (HCI) in the Bluetooth protocol stack presented in Figure 6. L2CAP was chosen because it introduces only a small overhead but still offers ordered and reliable data delivery. The only drawback is that the maximum packet size is limited to 673 bytes by default so additional segmentation and reassembly routines are required in order to transfer longer data chunks [13]. However, this is only a minor obstacle. Another possibility would have been to use TCP/IP but it was rejected because of the additional overhead introduced by the underlying RFCOMM, PPP and IP protocols.
3.1 PeerHood Library

PeerHood applications don’t use the daemon directly. Instead, they utilize the services provided by the PeerHood Library. Library’s class diagram is shown in Figure 7. Just like in the case of the daemon also the library’s architecture is based on plugins. Application developers using the PeerHood library see only the MPeerHood interface that contains all the functions they can use when coding their own applications. Possible additional components will also use the same interface.

3.2 PeerHood Interface

The main interface, MPeerHood, offers a set of functions that enable communication between applications and the PeerHood system. In this section some of the most important functions are described in more detail. These functions include e.g. the following (function parameters are omitted):

- **bool Init();**
  - Initializes the PeerHood instance. In short, these initialization routines include connection establishment between the library and the daemon.

- **TDeviceList* GetDeviceListL();**
  - Gets a list containing all nearby devices and their shared services and resources.

- **MAbstractConnection* ConnectL();**
  - Creates a connection to another PeerHood capable device. The connection endpoint is either the device itself or a shared service or resource located on the device.

- **bool RegisterService();**
  - Registers a local service so that other PeerHood capable devices can find and use it.

- **bool UnregisterService();**
  - Unregisters a previously registered service.

- **bool MonitorDevice();**
  - Sets a device under constant monitoring. If a change (out of range, back in range) takes place then the application is informed.

Whenever a PeerHood-capable device is found by the plugin it is assigned a tag that explains the networking technology that can be used when communicating with the device. If the application wishes to connect to that device it calls the PeerHood library. The library extracts the tag and gives it to the static Factory object that returns a MAbstractConnection object created by the MAbstractCreator that recognizes the tag. In the case of Bluetooth the tag is “bt-base” while for Wireless LAN it could be e.g. “ph-wlan”. The application can use the returned MAbstractConnection object to communicate with the remote device.
Applications can use the functionality behind the PeerHood system just like any other library without any special knowledge of the underlying network etc. Listing 1 shows a very simple PeerHood application that prints a list of all nearby devices to the screen and exits right after that.

Listing 1. A simple PeerHood application.

```cpp
#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->GetDeviceList();
    for (TDeviceIterator i = list->Begin(); i != list->End(); ++i) {
        std::cout << "Found device " << (*i)->GetName() << std::endl;
    }
    return EXIT_SUCCESS;
}
```

### 3.3 Prototype Status

Currently the prototype implementation exists only for Linux iPaqs but it will be ported to the Symbian OS in the near future. The prototype is not totally finished yet as features like resource (memory, storage space etc.) sharing are missing. However, the prototype can well be used for testing and research purposes even at its current state.

Implemented features include:
- Device discovery
- Service discovery
- Connection establishment
- Data transmission between devices
- Device presence monitoring

Figure 8 shows a typical view of the test application. The figure illustrates a situation where a couple of devices with Bluetooth are detected and one of them (00:02:C7:0B:9C:02) has PeerHood enabled. The device offer a Chat service to other devices. Simply double-clicking the corresponding service name would connect to the service.

![PeerHood test application](image)

**Figure 8. PeerHood test application.**

### 4 Conclusions

In this paper an approach for peer-to-peer capable mobile communication environment was presented. This environment was based on a concept of personal trusted device and its operation on some very local communication environment (peer-to-peer neighborhood). In this early stage of the research the paper was focused on the design aspects of the environment and thus no performance results were presented. However, this concept was implemented by using mobile devices (iPaqs) with Bluetooth capability. The implementation was restricted to the Bluetooth communication technology as it allows limited neighborhood sizes. According to the implementation design and the actual implementation the mobile peer-to-peer neighborhood is a viable approach for the future communication. Other networking technologies than Bluetooth can be easily added to the concept and the use of PeerHood is quite flexible.
References