IEEE International Conference on Telecommunications (ICT), Bucharest, June 2001

An Experimental Implementation of Mobile IPv6 on a Personal Digital Assistant

Zhao-shu Zeng and Michel Barbeau

School of Computer Science
Carleton University
1125 Colonel By Drive
Ottawa, ON K1S 5B6 Canada

{barbeau, zszeng}@scs.carleton.ca

ABSTRACT

IPv6 is a new version of the Internet Protocol that was standardised by the Internet Engineering Task Force (IETF). Mobility support in IPv6 is presently being standardised by the IETF Mobile IP Working Group. IPv6 has been designed to overcome all the problems within IPv4 and to provide solutions for the next generation networks. Devices such as Personal Digital Assistants (PDAs) with IP support are widely used in mobile computing. In this paper, we describe our experimental implementation of Mobile IPv6 on an PDA. The basic functions of a Mobile Node (MN) such as movement detection and registration have been implemented in this experiment.

I. INTRODUCTION

Today, mobile computing is becoming much popular with the increasing use of variety wireless devices offering IP connectivity, such as PDAs. When a node moves to another subnet, it expects to receive the same set of services regardless of the current location. As the IP protocol currently used (IPv4) can't make mobility transparent to the users, Mobile IPv4 was designed to solve this problem by using two IP addresses: home address and care-of address, for an MN. However, IPv6 is the trend for the future. It has the advantage of built-in mobility support. There is a movement in the wireless industry towards IPv6. For instance, the Third-Generation Partnership Project (3GPP) [16] has adopted IPv6 for their next generation of wireless network specification.

Palm OS is the operating system of a widely spread PDA architecture [11]. A problem of Palm OS is the lack of mobility support in its Net Library [11]. As the source code of Palm OS is closed, it is impossible to install or embed a new Mobile IP protocol in it. So we decided to develop our own protocol stack on Palm OS. Though the mobility support is possible for both Mobile IPv4 and Mobile IPv6, the mobility support has been integrated more efficiently in Mobile IPv6 than to Mobile IPv4. So we selected Mobile IPv6.

We implement a subset of Mobile IP operations in an IPv6 network, using a protocol development framework called p-kernel [13].

In Section 2, we present an overview of the mobility support aspects of Mobile IPv6. In Section 3, we give an overview of the p-kernel framework. In Section 4, we describe our Mobile IPv6 implementation on Palm OS. We present conclusions in Section 5.

II. OVERVIEW of MOBILE IPv6

The problem addressed by the Mobile IP Protocol is mobility of nodes, from network to network. The problem is not solved by *plain* IP because the IP address of a computer is bound to the network to which it is attached. Migration and attachment to a different network lead to the non-reception of packets, unless the MN's address is changed.

The Mobile IP approach relies on two-level addressing [12]. The MN has a long term IP address (called the home address) and a short term IP address for locating it when it is away from home (called the care-of address). Location Directories (LDs) store tables of associations of home address and care-of address. Tables are indexed by home addresses and updated according to the movements of MNs.

The point of attachment and care-of address of an MN may change while the home address remains fixed and connectivity is maintained.

A strength of Mobile IP is compatibility with normal IP. Mobile IP uses neither a specific IP address format nor an address range. Authentication protects MNs from remote redirection attacks. Low overhead was a design target so it can run over low bandwidth and high error rate wireless links and minimises use of battery power of mobile devices.

The main design concepts of Mobile IP are as follows. An MN is a computer that changes its point of attachment from one (sub) network to another.

A home network is a network identified in a home IP address.

A foreign network is a network to which an MN is momentarily attached and is given its short-term care-

of address.

Every MN is configured with its home address. Mobile IP provides a mechanism for dynamically obtaining a care-of address. The MN appears to the other computers of the Internet as if it is located in its home network. When the MN is in its home network, packets are delivered to the MN as usual. When the MN is in a foreign network, packets are readdressed and forwarded to the care-of address. Tunnelling is used for the readdressing and forwarding process.

A Home Agent (HA) is a router in the home network of an MN that is intercepting and tunnelling packets to the MN when it is not attached to its home network. It normally has an interface on the home network of the MN, for interception of packets.

IPv4 does not provide support for mobile communications. The next generation Internet protocol (IPv6) has built-in mobility support [7].

IPv6 addresses autoconfiguration of the care-of address of an MN when it attaches to a new link. Using the Neighbour Discovery protocol [8], [7], an MN is able to find the network prefix at any point of attachment it might select and then forms a globally routable IPv6 address for that point of attachment. When an MN moves, it informs its HA and Correspondent Nodes (CNs) about its new address, using the IPv6 Destination Option Header to carry the location information. Such option can be used with any data packet the MN sends. Therefore, it is relatively low traffic overhead.

When an MN sends a packet while away from home, it keeps mobility transparent to its application software and transport protocols. It moves the home address to a Home Address option. It sets the IPv6 header's Source Address to its care-of address. The destination node restores the MN's home address to be the IPv6 header's Source Address before delivering the packet to the upper layer. This way, the MN not only keeps mobility transparent to its upper software, but also passes the packet through any router implementing ingress filtering [4].

When an CN sends out a packet to an MN, it uses a Type 0 Routing header [2] to set an IPv6 Routing Header, specifying the care-of address as the destination address in the IPv6 header and the MN's home address as the final destination of the packet in the routing header. When the MN receives the packet at its care-of address, the MN processes this Routing Header and delivers the packet to the upper layer using the MN's home address as if the MN was at home.

III. OVERVIEW of the P-KERNEL

Most network software are structured into layers. Each layer contains a certain number of protocols. Network software is normally implemented as a part of the kernel of an OS (e.g. the Linux kernel [14]) but not always and may be embedded within a user level application (e.g. x-kernel [5]).

Palm OS is the operating system of a widely spread PDA architecture [11]. There is a need for an environment for doing research on mobile and wireless network protocols on the Palm OS, i.e. experimentation of new protocols at the physical, link, and network layers. The Net Library available for the Palm OS only supports a fixed set of protocols and does not allow integration of new protocols because it is not an open source. It is not usable as a vehicle for experimenting with the new mobile and wireless protocols such as Mobile IP.

Hereafter, we discuss a framework, the p-kernel, that we have been developing for implementing mobile and wireless network protocols [13] on PDAs. The framework is configurable and allows integration of a protocol graph which executable version runs as a component of a communication application on the Palm OS.

The p-kernel is a framework. It means that it provides a set of data structures, subroutines, a control loop, and slots in which a developer can hook its own data structures and subroutines in order to actualise the framework according to its needs. The needs correspond to a graph in which specific protocols such as Mobile IP, UDP, and SLP appear.

The p-kernel is inspired of the x-kernel [5] of which is adopted the object based model. There are three types of abstract objects: protocol, session, and message.

A protocol is a static entity, created when a protocol graph is instantiated, modelling a given protocol such as Mobile IP.

Protocols are composed of sessions that are created dynamically and modelling end-points of communication channels. A session maintains the state of the channel end-point, e.g. values of sequence numbers.

Messages are created dynamically and flow from one protocol/session to another. A message is normally made of a header part and a user data part.

An important concept within the model of x-kernel is the Uniform Protocol Interface (UPI). The communication primitives supported by protocols and sessions are uniformed and generic across all the layers. They are actualised by the protocol developer, using a function pointer mechanism. The UPI contributes to achieve a uniform structure from one layer to another that helps considerably to the clarity of a protocol stack.

An interesting feature in the implementation of this model is the message-per-process model. Indeed, each message, starting from its reception at the hardware level or its creation at the application level, is taken in charge by a thread. A protocol is implemented by a set of procedures. The procedures implement tasks of the protocols and sessions the associated message has to go through. A thread calls successively the procedures.

Concurrent threads may have to synchronise together. For instance it may be necessary to synchronise a thread handling an acknowledgement message with threads handling data messages and waiting for empty slots in a transmission window. Inter threads synchronisation is achieved using semaphores.

The x-kernel supports the aforementioned model on full scale computers. The challenge of p-kernel is to support that model on PDAs that have low memory, are slow, and have few communication resources. In addition, the Palm OS does not provide all the system programming concepts that we found on a full scale OS. For instance, there are no notion of thread and semaphore. It makes support of the x-kernel model on the Palm OS a little harder to achieve.

The p-kernel is implemented as a library that is embedded into an application that needs communication. A protocol graph is defined programmatically. When the application is entered, the protocol graph is instantiated, the p-kernel OS is created, and every protocol of the graph is initialised.

A typical application on the Palm OS is driven by a event processing loop. Execution of the application is triggered by events, e.g. a user interaction, arrival of a message. The association between the application and a p-kernel instance is made within the event loop of the application, as illustrated by the following pseudo code:

```
App_Event_Loop()
do {
EvtGetEvent(event);
<Give control to p-kernel>
      <Usual event processing of
      an application>
} while (event!=StopEvent);
```

When an event occurs, function `EvtGetEvent()` returns with an element of data representing the event. The control is first given to the p-kernel OS which either handles the event, if it has to do with telecommunications, or returns, otherwise. In the latter case, the event is handled according to a usual event handling model of a Palm OS application.

IV. OVERVIEW of the IMPLEMENTATION

We built an Mobile IPv6 stack using the p-kernel on Palm OS. The PDA works as an MN.

A. Test Application

For the purpose of this experiment, we built a communication application over Palm OS using the p-kernel. This application has a simple message transmission and reception user interface.

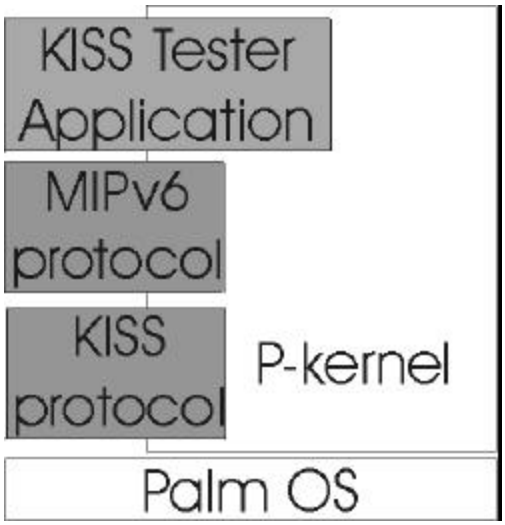

Figure 1. Logical model on an application

The application embeds a stack of protocols constructed using p-kernel. The stack contains two layers: KISS and MIPv6. KISS stands for Keep It Simple, Stupid [1]. It is a protocol implementing the AX.25 protocol [15]. It is used for passing and receiving frames over an RS-232 interface to a data radio.

Figure 1 pictures the logical model of the application built using the p-kernel. P-kernel is written in C++ and runs above the Palm OS. Protocols, such as the KISS and MIPv6 protocols, are implemented in C++ within the p-kernel framework. Applications, such as KISS Tester, are implemented using the p-kernel framework and any protocol implemented within it. The application is written in C.

The application is called KISS Tester and is pictured in Figure 2. The screen is divided into two areas. The upper area, in which data contained in incoming packets are posted. The lower area, in which data entered by the user is sent in packets.

Figure 2. User interface of the application KISS Tester.

B. Design of the MN

Movement detection and care-of address registration are the MN's basic functions we implemented in our experiment.

An MN uses Router Solicitation messages and Router Advertisement messages to detect its movements [10]. An MN discovers new routers and on-link subnet prefixes from Router Advertisement messages. An MN may send Router Solicitation messages or may wait for unsolicited (periodic) Router Advertisement messages to detect its movement. We choose to send Router Solicitation messages actively.

When an MN receives a Router Advertisement message, it modifies its default router list that records the current on-link routers' address, prefix information, and the lifetime of routers' entries. If it is a new router, it adds a new router entry in the router list. Furthermore, if it includes a new prefix, that means that MN enters a new link. The MN will form a new care-of address according to the new prefix information, register it with its HA, and notify its CNs.

Router entries have lifetimes. Timers are used to check lifetimes. If the router entry's lifetime expires, the router's entry is deleted. If there are no other routers with the same prefix, in the default router list, which the care-of-address is based on, it means that the MN moves out the network. The node deregisters the care-of-address with its HA and notifies its CNs.

An MN maintains a binding list of the nodes which it is currently communicating with, such as an HA and CNs. An MN registers or deregisters its care-of-address with the HA in its binding list using a packet containing Binding Update option. The MN will continue to send the packet with Binding Update option to its HA until it receives a packet with matching Binding Acknowledgement option or moves to a new link. A packet with Binding Update option is also used to inform the care-of address to CNs in the binding list. When an MN receives a packet including Binding Request option, it replies to the sender a packet including Binding Update option. This mechanism makes an HA capable of intercepting the packets addressed to an MN's home address and tunnel them to the MN. Moreover, the CNs that tend to send packets to the MN will use the MN's care-of address as the Destination Address in IPv6 header.

When an MN boots or detects that it has moved to a new link, it forms a new IPv6 address using IPv6 stateless address autoconfiguration [9]. The address is formed by combining a 64-bit address prefix with a 64-bit interface identifier [10], [3]. In this case, we use the data radio's call sign to create an interface identifier with IEEE EUI-64 format [6].

As a testing environment we use Linux workstations as Access Points (APs), routers, and an HA. They are set up for IPv6 support. In addition APs are set up for KISS and AX.25 support. As the AX.25 layer, under Linux, only supports IPv4 at the moment, we use Simple IPv6 Transition (SIT) to encapsulate IPv6 packet in IPv4.

At the moment, the MN on Palm OS can send out Router Solicitation messages and receive Router Advertisement messages from routers and an HA.

V. CONCLUSION

To cope with the complexity of implementing Mobile IPv6 on a small PDA, we adopted an incremental approach. We currently have a partial implementation of Mobile IPv6 on Palm OS. So far, we concentrated our efforts on control aspects of Mobile IPv6. We have implemented the following basic functions: router solicitation, router advertisement, binding update, binding request, and binding acknowledgement. Our testing application, including our protocol stack, has a footprint of around 30K bytes and runs on a Palm handheld.

We encountered difficulties during the implementation and testing phases. For the implementation, we use CodeWarrior for the Palm Computing Platform R6. This compiler is not completely ANSI C compliant. For example, Console-based I/O (the most commonly mentioned standard lib functionality) is not supported on this platform because the Palm OS itself does not support console I/O [17].

Memory usage is a main issue in Palm OS programming. Re-locatable memory allocated to a variable must be locked to avoid writing or reading errors. As Palm OS has limited memory space, we have observed that applications with large segments do not perform well when large contiguous blocks of memory are not available.

For the fixed network part of our testing set up (i.e. router, HA), we used Linux with experimental IPv6 support and AX.25 support. The Linux kernel with the experimental IPv6 module is not totally stable at this time and there are conflicts between different versions of the kernel and AX.25 tools. We tried several combinations before getting a stable and working environment. Besides, numerous tests were required to obtain successful transmission over our data radios. Of course, considerable debugging efforts had to be invested, not only for our program on Palm OS, but on Linux as well. Indeed, we had to trace the flow of packets in the IP stack of the Linux kernel, from the data link layer (AX.25) up to the IP layer, to determine for instance why some packets were dropped.

Data transfer is not supported in our implementation. Palm OS has its own Net Library which can be used to implement data transfer. This library, however, only supports IPv4 and is not open. It is not possible to add Mobile IPv6 in it. Therefore, it can be used to support data transfer as long as we use IPv4-compatible IPv6 addresses and IPv6 in IPv4 encapsulation. This approach is not an ideal solution because it increases the overhead. A better solution is

to not rely on the Net Library and to develop an entire IPv6 routine.

In a near future, we will enrich the Mobile IPv6 protocol elements supported by our implementation. Control mechanisms, such as dynamic HA address discovery, will be considered. We also need to focus on data transfer and implement new features such as reception of packets using IPv6 de-capsulation and transmission of packets with the Home Address option.

REFERENCES

[1] Chepponis, M. and Karn, P., The KISS TNC: A Simple Host-to-TNC Communications Protocol, in: Proceedings of 6th Computer Networking Conference, Redondo Beach, California, August, 1987, pp. 38-43.

[2] Deering, S. and Hinden, R., Internet Protocol Version 6 (IPv6) Specification, The Internet Engineering Task Force, Network Working Group, Request for Comments 2460, December 1998.

[3] Deering, S. and Hinden, R., IP Version 6 Addressing Architecture, The Internet Engineering Task Force, Network Working Group, Internet Draft, October 2000

[4] Ferguson, P. and Senie, D., Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing, The Internet Engineering Task Force, Network Working Group, Request For Comments 2267, January 1998.

[5] Hutchinson, N.C. and Peterson, L.L., The x-Kernel: An Architecture for Implementing Network Protocols, IEEE Transaction on Software Engineering, Vol. 17, No. 1, January 1991, pp.64-76.

[6] IEEE Guidelines for 64-bit Global Identifier (EUI-64)Registration Authority, January 2001, http://standards.ieee.org/regauth/oui/tutorials/EUI64.html

[7] Johnson, D.B. and Perkins, C., IETF Mobile IP Working Group, Mobility Support in IPv6, IETF Mobile IP Working Group, Internet-Draft, November 2000.

[8] Narten, T., Nordmark, E., and Simpson, W., Neighbor Discovery for IP Version 6 (IPv6), The Internet Engineering Task Force, Network Working Group, Request for Comments 2461, December 1998.

[9] Narten, T. and Thomson, S., IPv6 Stateless Address Autoconfiguration, The Internet Engineering Task Force, Network Working Group, Request for Comments 2462, December 1998

[10] Narten, T., Neighbor Discovery and Stateless Autoconfiguration in IPv6, IEEE Internet Computing, Vol. 3, Issue 4, July-August 1999, pp. 54-63.

[11] Palm Inc., Palm OS Platform, http://www.palmos.com , 2001

[12] Perkins, Charles E., Mobile IP Design Principles and Practices, Addison-Wesley, 1998.

[13] Robinson, S. and Barbeau, M., The p-kernel: A framework for mobile and wireless network protocol implementation on personal digital assistants, 2000. (available at: www.scs.carleton.ca/~barbeau)

[14] Satchell, S.T. and Clifford, H.B.J., Linux IP Stacks Commentary, CoriolisOpen Press, 2000.

[15] TAPR, AX.25 Amateur Packet-Radio Link-Layer Protocol, Version 2.2, November 1997 (available at: http://www.tapr.org.)

[16] Third-Generation Partnership Project. 3GPP – a global initiative. http://www.3gpp.org, 2001.

[17] Metrowerks CodeWarrior MSL C Reference, Metrowerks Inc. (available at CodeWarrior for Palm OS R6 CD).