INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

**ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600**

UMI[®]

An Experimental Implementation of Mobile IPv6 on a Personal Digital Assistant

By

Zhao-shu Zeng

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Master of Computer Science

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario

August 19, 2001

© Copyright
2001, Zhao-shu Zeng

**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**395 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**395, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-66873-8

Canada

The undersigned hereby recommend to
the Faculty of Graduate Studies and Research
acceptance of the thesis,

An Experimental Implementation of Mobile IPv6 on a Personal
Digital Assistant

submitted by

Zhao-shu Zeng

in partial fulfilment of
the requirements for the degree of
Master of Computer Science

Dr. Frank Dehne
(Director, School of Computer Science)

Dr. Michel Barbeau
(Thesis Supervisor)

Carleton University
August 19, 2001

Abstract

IPv6 has been designed to overcome some problems within IPv4 and to provide solutions for the next generation networks. Mobility support in IPv6 called Mobile IPv6 is presently being standardised. Mobile IPv6 is intended for the mobility support in the third-generation wireless data networks. Personal Digital Assistants (PDAs) are widely used in mobile computing, and Palm OS is the de facto OS standard on PDAs. However, there is no IP mobility support on the Palm OS. In addition, it is not allowed to install or embed new protocols in the Palm OS. Therefore, a subset of Mobile IPv6 implementation has been experimented in the user space of the Palm OS. Movement detection and registration of Mobile Nodes (MNs) have been implemented. Specifically, the following operations are included: router solicitation, router advertisement, binding update, binding request, and binding acknowledgement.

Acknowledgments

I would like to thank Professor Michel Barbeau, my supervisor, for his valuable advice, encouragement, responsiveness and patience. He has given me a learning experience for which I really appreciate.

I want to thank Mr. Javier Goviea and Ms. Jing Peng, for their kindly and generous help.

I am grateful to my family for their support and understanding. Thanks to my parents, Hui-yang Zhang and Guang-sen Zeng, for their encouragement and support all through my life. Special thanks to my husband, Xiao-fang Yuan, for his sharing with me all I have gone through.

Contents

Abstract	iii
Acknowledgments	iv
1 Introduction	2
1.1 Motivation	2
1.2 Problem Statement	4
1.3 Summary of Results	5
1.4 Thesis Outline	5
2 Review of Literature	7
2.1 Wireless Data Network	7
2.2 All IP-Based Network	8
2.3 Existing Solutions to IP Mobility	9
2.3.1 Mobile IP	9
2.3.2 Cellular IP	12
2.3.3 Hierarchical Mobility Solution	14
2.4 Overview of IPv6	14
2.4.1 Enough Address Space	15
2.4.2 Improved Header Processing	16
2.4.3 Neighbor Discovery	18
2.4.4 Plug and Play	18
2.4.5 Mobility Support	19
2.4.6 Built-in Security	19

2.5	Overview of Mobile IPv6	20
2.5.1	Mobile IPv6 Messages	20
2.5.2	Mobile IPv6 Operations	21
2.5.3	Mobile IPv6 Implementation Status	24
2.6	KISS TNC	25
2.6.1	Packet Radio	25
2.6.2	KISS TNC	27
3	Palm OS and P-kernel	29
3.1	Palm OS	29
3.1.1	Palm OS Platform Structure	30
3.1.2	Palm OS Memory Usage	31
3.1.3	Palm OS Network Library	31
3.1.4	Communication Connections and Services	32
3.1.5	Programming on Palm OS	33
3.2	P-kernel	34
4	Design of Mobile Nodes	39
4.1	Motivation	39
4.2	Software Structure	40
4.3	Software Scope	41
4.4	Scenario Descriptions	42
4.4.1	Movement Detection	42
4.4.2	Binding Management	43
4.4.3	Address Autoconfiguration	45
4.5	Data Structure	45
4.6	Activity Diagram	48
5	Implementation and Experiment	63
5.1	Application Structure	63
5.1.1	Test Application	65
5.1.2	IPv6 Layer	67

5.2	Testing Environment	73
6	Conclusion	76
6.1	Summary	76
6.2	Future Work	78
A	Publications	80
	Bibliography	81

List of Figures

2.1	Mobile IP Scenario	11
2.2	A Hierarchical Mobility Solution	15
2.3	Registration	22
2.4	Route Optimization	23
2.5	TNC Connected With PC and Radio	26
2.6	KISS Frame	27
2.7	TNC in KISS Mode	28
3.1	Palm OS Platform Structure.	30
4.1	Software Structure	40
4.2	MN's Use Case Diagram	44
4.3	IPv6 Unicast Address	46
4.4	Activity Diagram Symbols	49
4.5	Main Diagram	50
4.6	Receive Msg Diagram	51
4.7	Process RA Diagram	52
4.8	Detect Movement Diagram	54
4.9	Register Diagram	55
4.10	Binding Update Diagram	56
4.11	Process BA Diagram	58
4.12	Process BR Diagram	59
4.13	Router Tout Diagram	60
4.14	BU Timeout Diagram	61

4.15 Binding Tout Diagram	62
5.1 Protocol Stack	64
5.2 Open Sessions	65
5.3 Application Interface	66
5.4 Message Transmission Flow	66
5.5 Message Reception Flow	67
5.6 IPv4-Compatible IPv6 Address	74
5.7 IPv6 Encapsulated in IPv4	74
5.8 Test Environment	75
5.9 Registration Test	75

Contents

Chapter 1

Introduction

1.1 Motivation

Nowadays, many people use handheld wireless devices to access the Internet. For instance, people may use cellular phones based on WAP [31] to access the Internet to do such things as checking email and browsing web pages. The third-generation wireless data networks will make a significant contribution to the wireless Internet; that is, the convergence of two network technologies: wireless data communications and the Internet.

The Internet Protocol (IP) is dominant within the Internet system. The third-generation wireless data networks will provide IP access to Mobile Nodes (MNs). The homogeneous IP-based wireless access network technologies are being defined by wireless industries. For example, the Third Generation Partnership Project (3GPP) [30] and Mobile Wireless Internet Forum (MWIF) [12] are aimed at the design of third-generation networks with an IP-mobility solution.

Mobile and wireless devices, such as cellular phones, pagers, and personal digital assistants (PDAs), are widely used to get information. In the third-generation wireless data networks, wireless devices will have IP services as an integral part.

In brief, the third-generation wireless data networks will be IP-based. The mobility support functions will be handled in the IP layer.

Normal IP-based network access has a problem with the roaming of mobile devices. For instance, someone may want to move the mobile device from one subnet to another, and still receive the same set of services regardless of the location. In this case, communications can not continue until the device is configured with a new IP address, a new netmask, and a new default router IP address.

The problem lies in the routing mechanisms that are used in the current Internet. The currently used TCP/IP protocols assume that any node has a fixed point of attachment to the Internet. In particular the IP address of a node identifies the link on which the node is attached. If a node moves location without changing its IP address, the packets destined for the node will be delivered to its old location. Current Internet routing protocols require a node to change its network address when it moves to a new location.

Mobile IP [28] is standardised by the Internet Engineering Task Force (IETF) to provide IP mobility support. Mobile IP allows a node to leave its home subnet and move to other subnet while remaining connected to the rest of the Internet. A mobile node (MN) is identified by its static home address. When the MN attaches to a new link it obtains a new IP address called care-of address. The MN informs routers in its home network about its new location. The packets arriving at its home address are tunneled to its care-of address.

Mobile IP handles all mobility-related functions at the IP layer. This mechanism transforms the mobility problem to a routing problem. In addition, it makes mobility transparent to the upper protocols and applications.

There are two versions of Mobile IP: Mobile IPv4 and Mobile IPv6 [20]. Mobile IPv4 is an extension of IPv4 with support for mobility. Since IPv4 is the currently used IP version, the implementation of Mobile IPv4 is compatible with the current TCP/IP networks. Therefore the TCP/IP infrastructure does not need to change to support mobile computing. However, IPv4 was defined in the 1970s. The rapid development

of the Internet has brought more and more new technologies and more and more users. IPv4 has some problems in meeting the requirements of today's Internet. One of the well known problems is the address space. IPv4 uses a 32-bit address space to provide over 4 billion addresses. The IPv4 addresses are not assigned with 100 percent utilization because of the interaction of routing and addressing mechanisms. The depletion of the seemingly large number of IPv4 addresses will happen in the near future [23]. Another problem lies in the lack of security support in IPv4. Mobile IPv4 relies on its own security mechanisms that are based on statically configured *mobility security associations*.

IPv6 [9] has been defined to solve some of the problems of IPv4. It offers a 128-bit address space to preclude running out of unique addresses in IPv4. Another upgrade in IPv6 is the support of address autoconfiguration. A new node can autoconfigure its address when it plugs into an IPv6 network. It is very convenient for MNs. When MNs move from one subnet to another, they can get their new address automatically. In addition, security support is a default in IPv6. Furthermore, an advantage of IPv6 is the built-in mobility support called Mobile IPv6. The integration of mobility support features in IPv6 makes Mobile IPv6 more efficient than Mobile IPv4. For example, some fields in IPv6 packet are already defined to carry mobility information. Thus Mobile IPv6 provides enhanced mobility transparency, built-in smooth handoffs, and built-in route optimization. There is a movement in the wireless industry towards IPv6. For instance, IPv6 has been adopted by 3GPP for their next generation wireless network specification.

1.2 Problem Statement

PDAs are widely used in mobile and wireless computing, and Palm OS is a widespread OS architecture for PDAs [19]. In the third-generation IP-based wireless data networks, wireless devices such as PDAs will provide IP services. However, there is no IP mobility support in the Palm OS. There are some problems to experiment with new IP mobility support protocols on the Palm OS. Firstly, it is impossible to install

or embed a new protocol in the Palm OS because the source of the Palm OS is closed. Secondly, although there is a net library to provide IP services in the Palm OS, it only supports IPv4. Thus it cannot be used to provide normal IP services when a new version IP (e.g., IPv6) is developed on the Palm OS.

1.3 Summary of Results

We have experimented with Mobile IPv6 implementations on the Palm OS. As stated in Section 1.2, the source of the Palm OS is closed, and the net library of the Palm OS cannot be used in this experiment. So a new wireless protocol stack is built in the user space. This stack includes wireless protocols in the network layer (e.g., Mobile IPv6) and the data link layer (e.g., AX.25 [3]).

A protocol development framework called p-kernel [33] is used in the implementation. The p-kernel provides a set of data structures, subroutines, and slots. It enables developers to embed new protocols. In addition, the p-kernel provides a Uniform Protocol Interface (UPI) and APIs. These interfaces hide layer interactions and Palm OS programming details from developers.

Above all, a subset of Mobile IPv6 operations is implemented on the Palm OS, these being movement detection and registration. The movement detection lets an MN determine its new location. The registration enables the rest of the nodes on the Internet to be informed about the new location of an MN. Thus an MN is continuously reachable during its movement.

1.4 Thesis Outline

This thesis describes an experimental implementation of Mobile IPv6 on a PDA. It is structured as follows.

Chapter 2 is about the third-generation wireless data networks. It briefly reviews

the two IP mobility solutions: Mobile IP and Cellular IP. An overview of IPv6 and Mobile IPv6 is also presented. Moreover, the AX.25 protocol and KISS TNC that are used in this experiment are discussed briefly.

An overview of the Palm OS and the p-kernel framework is presented in Chapter 3. In this chapter, the Palm OS structure and its communication services and resources are described. The programming and memory usage model on the Palm OS are also introduced. As a protocol development framework on the Palm OS, the structures and main objects of the p-kernel are detailed.

Chapter 4 describes the design of MNs that support Mobile IPv6 on the Palm OS. In this chapter, the design issues, software architecture, data structures, and activity diagrams are presented.

Chapter 5 gives the implementation and testing environment details.

Chapter 6 concludes the thesis with a suggestion for future extensions to the implementation.

Chapter 2

Review of Literature

2.1 Wireless Data Network

The fast growth of the Internet, combined with the widespread availability of highly mobile small hosts in the form of laptop and palmtop computers, and other PDAs, has created a big demand for the concept of mobile computing. The current trends are clearly towards further miniaturization and greater mobility. It is expected that many of tomorrow's hosts will be in the form of pagers, handhelds, and cell phones. In addition, they will access the Internet via the wireless data networks.

Current wireless data services based on circuit switched radio transmission do not satisfy the users and providers. A dedicated channel is assigned to a single user during the entire call period, therefore the service is costly and the connection setup time is long. Obviously, the utilization of the traffic channel is inefficient and data rates are low. Wireless packet switched services have been developed to address these inefficiencies; because a channel is assigned only when it is needed and released after the transmission is finished, packet switching is especially efficient for bursty traffic and multiple users.

The General Packet Radio Service (GPRS) [4] is a packet switched wireless data

network architecture currently being defined by the European Telecommunication Standards Institute (ETSI). It offers packet data services using the Global System for Mobile Communications (GSM) cellular networks. The GPRS mobile stations route data packets directly to external packet data networks, such as IP networks and X.25 networks. 3GPP is developing the next generation of wireless network standards based on GPRS for GSM-based systems.

2.2 All IP-Based Network

Today IP is the dominant networking protocol. It forms the basis of backbone wired data networks. Mobile IP is a component of IP to provide mobility support in the IP layer [29]. With the wireless packet data network technology, the convergence of IP and mobile telecommunications in wireless networks is possible. This approach presents a number of challenges to the existing capabilities of Mobile IP and the third-generation networks. A number of standard bodies (e.g., IETF), companies and industrial forums (e.g., 3GPP) are defining the future IP-based mobile telecommunications networks.

IP-based networks bring a lot of benefits. First of all, applications of wired data networks can also operate on mobile and wireless networks. Secondly, integrated wired and wireless networks make management much easier. In addition, advantages of IP technology, such as IP telephony [14] and quality of service (QoS) (e.g., differentiated services [5]) can be directly used in the wireless data networks. This will enable wireless networks based on IP to provide voice services as well as data services. Furthermore, all mobility-related functionality can be handled in the IP layer. Thus mobility support is transparent to other layers. This enables the deployment of a homogeneous, IP-based network that is independent of the different wireless interfaces. Therefore the IP-based networks provide a good framework for scalable global mobility solutions.

2.3 Existing Solutions to IP Mobility

Networks are becoming pervasive and ubiquitous, which means that the network connectivity will be anytime and anywhere. Generally, there are two different kinds of mobility: user mobility, code mobility, and node mobility.

In user mobility, user moves and uses services from different computers with network access. User does not carry any terminal. There are several ways to solve the user mobility problem. One of them is called Session Initiation Protocol (SIP) [15]. SIP is an application-layer control (signaling) protocol for creating, modifying and terminating sessions with one or more participants. SIP supports user mobility by proxying and redirecting requests to the user's current location. Users can register their current location.

Code mobility [13] is defined as the capability to dynamically change the bindings between code fragments and the location where they are executed. It is aimed at providing a solution to the traditional Client/Server structure of distributed applications that is fixed and defined statically. Thus it addresses the requirements of service customization, dynamic extension of application functionality, and autonomy. Code mobility also can be used to support user mobility.

In node mobility, user moves carrying a node. To solve the node mobility problem, several frameworks such as Mobile IP and Cellular IP have been proposed to support seamless network access to MNs. Mobile IP addresses slow mobility while Cellular IP addressed fast mobility. The details of the two IP mobility protocols are presented hereafter.

2.3.1 Mobile IP

Mobile IP represents a solution to the mobility support problem. The problem addressed by the Mobile IP 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. Mobile IP is a modification to IP that allows nodes to receive packets continuously no matter where they attach to the Internet. Routing of packets destined for an MN will be changed so that the packets arrive at the MN's new point of attachment. This approach is quite simple because it transforms the mobility problem to a routing problem.

The Mobile IP approach relies on two-level addressing. An MN has a long-term IP address (called the home address) for identifying it 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 addresses and care-of addresses. 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 the connectivity is maintained.

A strength of Mobile IP is the compatibility with the normal IP. Mobile IP uses neither a specific IP address format nor an address range. An MN is able to communicate with other nodes that do not implement Mobile IP. 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 minimizes use of battery power of mobile devices.

The main design concepts of Mobile IP are as follows (Figure 2.1).

An MN is a computer that changes its point of attachment from one sub network to another.

A correspondent node (CN) is a computer that communicates with the MN.

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

A foreign network is a network to which an MN is temporarily attached and is given its short-term care-of address.

Figure 2.1: Mobile IP Scenario

Every MN is configured with its home address. Mobile IP provides a mechanism for dynamically obtaining a care-of address. An MN appears to the other computers of the Internet as if it was 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 re-addressed by changing the original destination IP address (the home address) to a different destination IP address (the care-of address) and forwarded to the care-of address. Tunneling is used for the re-addressing and forwarding processes. When the MN receives the packet at its care-of address, the inverse re-addressing is needed to keep the higher-level protocols transparent to mobility.

A Home Agent (HA) is a router in the home network of an MN. Whenever an MN acquires a new care-of address, it sends a packet to inform its HA and make sure the HA is aware of the care-of address. This process is called registration. The HA stores the MN's care-of address in its LD. This enables the HA to intercept and tunnel packets to the MN when the MN is not attached to its home network. The MN normally has an interface on the home network of the MN, for interception of packets.

Therefore, Mobile IP provides a framework for users roaming outside their home network who need to keep connectivity all the time. However, it is not designed for fast handoff management. The MN needs to inform its home agent whenever it moves. If it moves fast and frequently from one subnet to another, it needs to send

a message, including the new location, on every move. It takes time for the home agent to get this message. Until the home agent knows its exact current location, the packets addressed to the MN's old location will be lost. The worst thing is when the home agent is far away from the MN, since the loss and delay of packets to the MN will be significant. Also, sending network update messages frequently will result in excessive battery power consumption, which is unacceptable for portable small wireless devices.

In short, Mobile IP is a solution to IP mobility support, especially suitable for relatively slow movement and wide area mobility, such as movement between different administrative domains or geographical regions. In addition, Mobile IP makes mobility transparent to transport and other higher layers, so that the existing routing infrastructure, non-mobile hosts, and current applications, do not require modification.

2.3.2 Cellular IP

Although people rarely read text or watch video when they are driving or walking, they may want to download files or talk on the Internet phone while doing so. As mentioned in Section 2.3.1, Mobile IP is not a good solution for this kind of environment because it is optimized for slow movement across different subnets in the wide area. Cellular IP is addressed to wireless access networks and highly mobile hosts, that is, it is designed to support local mobility, say, fast movements between base stations in cellular networks.

Cellular IP is an improvement of the cellular network to fit the next generation all-IP based wireless networks. It inherits cellular principles, such as passive connectivity, paging, and fast handoff control, and uses them around the IP model. The network model of Cellular IP [7] is described as follows.

A base station serves as a wireless access point and a router of IP packets in an access network. Each base station has an IP address.

A gateway is a router via which the cellular access networks are connected to the Internet. The gateway router could work as a home agent for the mobile hosts that see this access network as their home network. Similarly, the IP address of the gateway router could also be used as an MN's care-of address when the MN is away from home and attached to the cellular network. Packets transmitted by mobile hosts are first routed to the gateway router and then to the Internet.

A mapping consists of two entries: an MN's IP address and base station ports that are used to forward the packets from the MN or to the MN. In Cellular IP networks, each base station does not need to know the exact location of MNs. The base station only needs to know which of its outgoing ports to forward packets for an MN. The mapping is remembered by the base station when the packets from an MN are routed to the gateway router on a hop-by-hop basis.

Paging means an idle MN periodically sends short control packets called paging-update packets to the nearest available base station. These packets are routed on a hop-by-hop basis and arrive at the gateway router. The base stations on this route monitor the passing packets and maintain the mapping in the cache. By monitoring these passing packets, the base station creates a hop-by-hop reverse path for future packets addressed to the MN.

As the MN moves, it keeps sending its paging-update messages to the nearest base stations, thus the base stations have up-to-date mappings of this MN. Outdated mappings are cleared after a system-specific timeout. When the new and old mapping coexist for a short time during the migration, these two mappings guarantee the MN is reachable and the on-fly packets will not be dropped. As the outdated mapping will be cleared after a period of time, if an MN leaves the access network, it does not need to inform the base stations about its leaving.

A Paging Cache is used to store the mappings of idle MNs. A Routing Cache is used for the mappings of active MNs, say, the MNs which are currently sending and receiving real data. The other difference between them is the timer to clear the outdated mapping. The timer for the Paging Cache is higher than that for the

Routing Cache. This is because the nodes in the access network only need to know the coarse location information for idle MNs. The separation of paging and routing cache results in a significant decrease in traffic load, especially when there are large number of MNs attached to a wireless access network, only a small percentage of which are receiving and sending data packets.

A benefit of Cellular IP is that it is fully compatible with IP. It uses control packets that can be implemented as a new IP option. Therefore it does not need to change the IP packet format or ask for extra address space. Because these packets never leave the wireless access network, regular IP routers do not need to be updated.

2.3.3 Hierarchical Mobility Solution

As mentioned in Section 2.3.1, Mobile IP provides a mobility solution in wide area mobility and meets the goals of operational transparency and handoff support. However, it lacks support for fast handoff control and real-time location tracking found in Cellular IP. Cellular IP has the advantage for rapid movement, but it is limited to supporting mobility in local cellular networks.

Significant efforts are presently underway to develop a comprehensive solution for transparent mobility in the Internet. To overcome the limitations of Mobile IP and Cellular IP, there is an approach of hierarchical mobility management. That is, the Cellular IP handles local mobility and uses Mobile IP for wide area mobility (Figure 2.2). By separating local and wide area mobility, the performance of mobility support can be significantly improved [40].

2.4 Overview of IPv6

IPv6 is designed to solve the problems of IPv4. These problems include: shortage of available IP addresses, lack of plug-and-play host configuration, lack of mechanisms to handle mobile computing, lack of network layer security, rapid growth of backbone

Figure 2.2: A Hierarchical Mobility Solution

routing tables.

Compared to IPv4, IPv6 provides the following new features: enough address space, header format simplification, improved support for extensions and options, neighbor discovery, plug and play, mobility support, and built-in security.

2.4.1 Enough Address Space

The main goal of IPv6 is to provide enough IP addresses. The 32-bit IPv4 addresses are expected to be used up in the near future. Network Address Translator (NAT) provides a solution by using only a small number of global addresses for thousands of nodes in a network. It intercepts all the packets leaving and arriving at the network and changes the private internal address into a global address, or vice versa. One of the serious issues with NAT is security, especially ensuring that a packet carried across a network is not changed by malicious nodes. For instance, because the address of a packet is changed by the NAT, the packet with IP security (IPsec) [21] standard is failed in the IPsec authentication check. Thus the packet will be dropped by the receiver. Though the deployment of NAT slows down the depletion of IPv4 address,

it does not solve the address space problem at all.

IPv6 has a 128-bit address space so it can offer addresses to all possible network appliances. It is necessary for next generation networks with widespread use of wireless devices, such as PDAs and cell phones.

IPv6 addresses are divided into three types: unicast addresses, multicast addresses, and anycast addresses. An anycast address is an identifier for a set of interfaces; a packet destined for an anycast address is delivered to one of the interfaces identified.

Within this huge address space, a tiny part is reserved for all current IPv4 unicast addresses, called IPv4-compatible IPv6 addresses and IPv4-mapped IPv6 addresses. Another part of addresses is reserved to represent addresses with different ranges of usage. They are link-local addresses, site-local addresses, and global-scope addresses. The link-local addresses are not routable but are guaranteed to be unique on a link. Nodes on the same link can communicate with each other even without any routers, by using their link-local addresses. The site-local addresses are unique only within a specific site or organization. They can only be used within a site and cannot be forwarded past site boundaries. The global-scope addresses are routable in the global networks.

IPv6 addresses are assigned to individual interfaces on nodes, not to the nodes themselves. A single interface may have multiple unique unicast addresses. This provides flexibility in network renumbering. The multi-homed host is easy to switch from one network to another when, for example, the Internet Service Provider is changed.

2.4.2 Improved Header Processing

With respect to IPv4, IPv6 header processing is improved by a simplified header format and the enhanced support for extensions and options. The unnecessary fields in the IPv4 header, such as Internet Header Length (IHL) and Header Checksum,

have been eliminated. Checksums are performed by higher protocols when they are needed. This reduces the work of a router when it forwards a packet because the router does not need to calculate the checksum.

IPv6 defines several kinds of extension headers, which could be used to include additional information in the IPv6 packets. The IPv6 extension headers include Destination Option header, Routing header, Hop-by-Hop Option header, Authentication header, Encapsulation Security header, and Fragmentation header. Each extension header is an integer multiple of 8 bytes long. Multi-byte fields within each extension header are aligned on their natural boundaries. This kind of alignment requirement is optimized for efficient parsing of packets. Furthermore, the options in extension option headers do not have any length limitation, though options in IPv4 are limited to a total of 44 bytes.

In IPv4, every IP option is treated as a Hop-by-hop option and thus causes performance degradation when an option only relates to the final destination node. IPv6 changes this mechanism. The Hop-by-hop Option header in IPv6 has the same function as that in IPv4. The Destination Option header in IPv6 is used to carry a sequence of one or more options to be processed only when the packet arrives at the final destination node. It is especially useful for carrying mobility support options, because the mobility information is only used by the destination nodes.

The Routing header in IPv6 is similar to the Source Routing header defined in IPv4, which can be used both as a strict source route and a loose source route. Mobile IPv6 uses the Routing header only as a loose source route. Unlike the IPv4 Source Route Option, the recipient of a packet with a Routing header is not required to use the same route when it needs to send packets back to the sender.

IETF developed the IPsec protocol [21], which is optional for IPv4 and a default in IPv6. Authentication and encryption are implemented as IPv6 extension headers. An IPv6 node can send and receive packets with encrypted contents, and the correct identity of its peers is assured.

2.4.3 Neighbor Discovery

The IPv6 Neighbor Discovery protocol [24] is used by the nodes on a common link to discover each other's presence, link-layer address, and reachability information. Also, it is used to find neighbor routers.

The Neighbor Discovery protocol uses two pairs of messages to accomplish its functions. Neighbor Solicitation (NS) and Neighbor Advertisement (NA) messages are defined to discover the link-layer addresses of neighbors and to verify that a neighbor is reachable. Router Solicitation (RS) and Router Advertisement (RA) messages are defined to locate and obtain information about routers.

The mechanism to verify a neighbor's reachability is called Neighbor Unreachable Detection (NUD). An NS message is sent to a neighbor whose link-layer address is already known. The neighbor returns a NA message to announce its reachability. If no messages are returned by the neighbor, the sender assumes that the neighbor is unreachable and then switches to another path.

A host sends RS messages to find routers actively. Routers send RA messages to advertise their presence with other configuration information, such as link prefix and maximum transmission unit (MTU). The link information carried in RA messages enables address autoconfiguration by hosts.

In brief, the Neighbor Discovery protocol in IPv6 combines, extends, and improves IPv4 Address Resolution Protocol (ARP), IPv4 Internet Control Message Protocol (ICMP) Router Discovery, and IPv4 ICMP Redirect.

2.4.4 Plug and Play

Plug and Play means that new machines can be added to a network by automatically configuring a usable IPv6 address and other network parameters such as prefix, router address, and the Name Server address. IPv6 Address autoconfiguration includes stateful and stateless address auto-configuration.

In the stateful autoconfiguration model, hosts obtain addresses and other configuration information from servers, such as Dynamic Host Configuration Protocol (DHCP) servers.

The stateless autoconfiguration model lets an IPv6 node form its address without administrative servers. When a node starts the bootstrap process, it could create a link-local address using its locally available information such as its own unique interface identifier. Almost all the network interfaces have embedded IEEE identifiers, such as Media Access Control (MAC) addresses of Ethernet interfaces. IEEE also defines a 64-bit Extended Universal Identifier (EUI-64) [2]. An EUI-64 based interface identifier can be created from an IEEE 802 48-bit MAC identifier [10]. IEEE identifiers are designed to be globally unique and could be used to form a unique IPv6 address. A host could create its link-local address with the 64-bit standard link-local prefix and its 64-bit interface identifier. Also, with the link information obtained in the RA messages, a host creates its global address by combining its interface identifier and link prefix information advertised by routers.

2.4.5 Mobility Support

IPv6 has been enhanced to address mobility requirements. Mobile IPv6 meets the following mobility support requirements: a globally unique home address assigned to every MN, an MN always addressable by its home address, an MN able to communicate directly with nodes that do or do not implement mobility support, handoffs transparent to mobile and correspondent nodes, and authenticated packets used to inform another node about the location of an MN. Mobility in IPv6 is discussed in more detail in Section 2.5.

2.4.6 Built-in Security

Security is a very important issue for the Internet. IPv6 uses IPsec for all security requirements, like authentication, data integrity protection, and replay protection. The

implementation of the IP Authentication Header is mandatory for the IPv6 nodes. It could provide a mechanism for wide-scale adoption of route-optimization techniques. It can protect the Routing header, Destination Option header, and tunneling under mobile situations.

2.5 Overview of Mobile IPv6

IPv4 does not provide support for mobile communications. Therefore Mobile IPv4 was defined to support mobility management in the network layer. Also, there is a new approach to IP mobility support, that is, built-in mobility in IPv6 called Mobile IPv6. Though Mobile IPv6 incorporates the principles of Mobile IPv4, such as home network, home agent, and care-of address, there are some differences between them. For example, there is no foreign agent in Mobile IPv6.

2.5.1 Mobile IPv6 Messages

Mobile IPv6 requires new messages to carry some additional mobility information. They are defined as the IPv6 Destination Options because they only need to be examined by the packet's destination node.

The following four new Destination Options are defined in the Internet Draft of Mobile IPv6 [20].

- Binding Update

The Binding Update option is used by an MN to inform its HA and CNs about its current care-of address. Any packet including a Binding Update option must also include either an Authentication Header or an Encapsulation Security Payload header. Binding Update also refers to a packet with a Binding Update option.

- Binding Acknowledgement

The Binding Acknowledgement option is used by the HA or CNs to acknowledge the receipt of a Binding Update, if an acknowledgement is requested. Any packet including a Binding Acknowledgement option must also include either an Authentication Header or an Encapsulation Security Payload header. Binding Acknowledgement also refers to a packet with a Binding Acknowledgement option.

- **Binding Request**

The Binding Request option is used by any node to request an MN to send a packet with a Binding Update option. Binding Request also refers to a packet with a Binding Request option.

- **Home Address**

The Home Address option is used by an MN to inform a receiver about the MN's home address. With this option, the MN can use its care-of address in the source IPv6 field of the header.

2.5.2 Mobile IPv6 Operations

Address autoconfiguration, registration, built-in route optimization, enhanced mobility transparency, built-in smooth handoffs, and home agent discovery are the main operations of Mobile IPv6.

Address Autoconfiguration

An MN can use RS and RA messages to determine its new location [20]. When an MN moves to a new link, it discovers new routers and on-link subnet prefixes from the received RA messages. Then the MN forms its new care-of address using IPv6 stateless address autoconfiguration [25], that is, combining the link prefix and the MN's interface identifier. Another way is to use IPv6 stateful address autoconfiguration, that is, the MN obtains the care-of address from the administrative servers

(e.g., DHCPv6 [6] servers).

Registration

After forming its new care-of address, the MN registers its care-of address with its HA. Therefore it sends a packet containing a Binding Update option to the HA. The HA returns a packet containing a Binding Acknowledgement option to the MN (Figure 2.3).

Figure 2.3: Registration

Route Optimization

When an MN registers its new care-of address to its HA, it also sends a packet with a Binding Update option to inform the CNs about its new care-of address. This allows CNs to cache the new care-of address and send packets directly to an MN with the destination address being the MN's current care-of address. This way the triangle routing is avoided and the route optimization is accomplished (Figure 2.4).

When an CN sends out a packet to an MN, it uses a Type 0 Routing header [9] 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

Figure 2.4: Route Optimization

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. Conversely, in Mobile IPv4, encapsulation is used to send any packet to an MN, which causes increased packet overhead and waste of bandwidth.

Enhanced Mobility Transparency

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 [11].

Built-in Smooth Handoff

As IPv6 nodes could have multiple IP addresses per interface, it is possible for an MN with overlapping cells to accept packets destined for its previous care-of address, even after registering a new care-of address with its HA. Hence the MN is able to maintain simultaneous connectivity at the old care-of address and the new care-of address.

In cases when an MN cannot maintain both an old care-of address and a new care-of address when it moves, the previous router will assist the MN to continue to accept packets sent to the old care-of-address. The MN sends a Binding Update to the router servicing the previous care-of address. Thus the packets are delivered to the new care-of address by the previous router. The previous router operates in the same way as when the MN's HA receives the Binding Update from the MN. That is, the previous router intercepts any packets destined for the home address in the Binding Update (the previous care-of address), and tunnels them to the care-of address indicated in the Binding Update (the new care-of address).

HA discovery

When an MN is away from home, things may happen such as the router that acted as its HA being replaced by a different router. MNs can dynamically discover the address of its HA by sending an ICMP Home Agent Address Discovery Request message to the Mobile IPv6 Home-Agents anycast address on their home subnet. The anycast address is configured by all HAs on the subnet. This method makes HA discovery work well even in cases where some of the HAs are down.

2.5.3 Mobile IPv6 Implementation Status

Though the definition of the standard of mobility support in IPv6 is still in progress, some organizations and universities are working on the implementation of Mobile IPv6.

Linux OS already has a working, but not complete, IPv6 implementation in the 2.2.* and 2.3.* series kernels. Some of the IPv6 options have been implemented. They are linked to current kernel source. Since the Linux OS is an open OS and a clone of UNIX, most of the research about implementation of Mobile IPv6 is done on Linux [26] [27]. In addition, there are implementations on SUN's Solaris OS [22], FreeBSD [39], and WINDOWS NT [32].

Mobile computing is a main function for PDAs. Palm OS is the principle OS of widely used PDAs. However, research of Mobile IPv6 implementation on the Palm OS was not found at the time of writing. Therefore, there is a need to explore the implementation of Mobile IPv6 on the Palm OS.

2.6 KISS TNC

2.6.1 Packet Radio

Packet radio [37] is a particular digital mode of Amateur Radio communications. It takes any data stream sent from a computer and sends that via radio to another amateur radio station similarly equipped. Packet radio uses packets to transmit through radio. It links Amateur Radio stations together directly or by means of a network.

An Amateur Radio station is made up by three elements: a Terminal Node Controller (TNC), a computer or terminal, and a radio (Figure 2.5) [38].

A TNC is used to interface a computer to a radio [41]. It is basically a Packet Assembler/Disassembler (PAD), with a built-in modem to convert outgoing digital bit streams into audio tones suitable for modulating the radio transmitter, and to convert incoming audio from the radio receiver into digital data for the PC. It is typically connected to a host computer or terminal with an RS-232 Data Communications Equipment (DCE) connector. It implements all necessary channel protocols in an

amateur packet radio network, since they are difficult to implement on personal computers which do not provide real-time multitasking operating systems. This approach allows the use of packet radio in a much wider variety of applications.

A computer or terminal is the user interface. It runs a terminal emulator program.

A radio in different frequency is used to transmit different speed packets.

Packet radio can support multiple conversations on the same frequency at the same time. It uses a protocol called AX.25 [3] to accomplish this shared channel. AX.25 is a standard for a link level protocol. It is designed to be used between two Amateur Radio stations in a point-to-point communication environment or to carry traffic of other protocols such as TCP/IP. The link protocol is based on, and very close to High-level Data Link Control (HDLC).

The most interesting feature of the Amateur Radio packet network is the addressing scheme. Each radio amateur has a unique callsign, which is a string of 4 to 6 letters and digits that is assigned by the government. Every AX.25 packet sent with the sender's and recipient's amateur callsign, thus providing the station identification with every transmission.

Figure 2.5: TNC Connected With PC and Radio

2.6.2 KISS TNC

The Keep It Simple, Stupid (KISS) protocol is used to communicate between a PC and a TNC. It is a very simple asynchronous packet protocol, whose main purpose is to form AX.25 frames and provide an envelope for them.

Figure 2.6 shows the KISS frame format. Each KISS frame starts and finishes with a Frame End (FEND) character. Immediately following the leading FEND character is a KISS type byte. Following the field of the KISS type is the AX.25 frame. There is no checksum or CRC.

Figure 2.6: KISS Frame

When a TNC operates in KISS mode [8], almost all of the station's functionality takes place within a PC (Figure 2.7). The PC provides the high-level network services such as file transfer protocol (ftp), remote terminal connection (telnet) and so on, together with the lower level protocol software which implements the protocols of the transport layer (e.g., TCP, UDP), Internet layer (e.g., IP), and data link layer (e.g.,

Figure 2.7: TNC in KISS Mode

KISS).

A TNC in KISS mode includes the implementation of the KISS protocol that converts a KISS frame to an AX.25 frame, or vice versa. Thus the KISS frame is sent between the PC and the TNC, and the AX.25 frame is sent between the TNC and the radio (Figure 2.7). The AX.25 protocol is removed entirely from the TNC, which makes the TNC adaptable to different format of frames without reprogramming.

Chapter 3

Palm OS and P-kernel

3.1 Palm OS

Computers and the Internet provide more information to people than ever before in history. However, people also want to get information freely, that is, at anytime and any location. So, it is practical for people to have a handheld computer in a pocket rather than a desktop or laptop and to have access to information whenever and wherever they need it.

The Palm OS is the de facto operating system standard for handheld computing, such as PDAs. It aims at simplicity, wearability and mobility. Compared with desktop computers, PDAs have smaller memory (e.g., 1MB, 4MB or 8MB), slower processors (e.g., 16 MHz Motorola 68000 type processor), and fewer communication resources (e.g., one serial port). So the Palm OS does not support all system programming concepts like in a full scale OS. For instance, it only supports single thread applications and no semaphore library.

Figure 3.1: Palm OS Platform Structure.

3.1.1 Palm OS Platform Structure

The Palm OS platform consists of five primary components: Palm OS software, reference hardware design, HotSync conduit data synchronization technology, platform component tools including an API that enables developers to write applications, and software interface capabilities to support hardware add-ons.

Figure 3.1 pictures the architecture of Palm OS. Since PDAs are designed for users on-the-go, device applications must be loaded without delay and easy to use. Device applications include simple personal information management (PIM) applications, such as Address book and To Do list. The 3rd party applications include web clipping applications, which are HTML-based content designed specifically for a small form-factor and wireless devices. Web clipping applications allow web sites to deliver compressed information to users in a short time, such as under 10 seconds on average. Other System libraries let developers enable extensions to the functionality of the Palm OS. For instance, a TCP/IP library could be loaded to support network services.

The hardware abstraction layer is an internal software map of how the hardware works. It gives developers the flexibility to modify the Palm OS without having to rewrite code each time a hardware change is made.

3.1.2 Palm OS Memory Usage

A handheld Palm OS device has limited heap space and storage space. For example, at the time of writing, the total size of dynamic memory and storage available in different versions of the device is between 512KB and 8MB. The device has neither a disk drive nor PCMCIA support. Application optimisation is critical because of the limited space and power. For example, applications with large segments do not perform well when large contiguous blocks of memory are not available. To make applications fast and efficient, the strategy is to optimize for heap space first, speed second, and code size third [19].

The total available RAM storage is divided into two logical areas: dynamic RAM and storage RAM. Dynamic RAM is analogous to the RAM in a typical desktop system. It is used as the working space for temporary allocations. Storage RAM is the remainder of the available RAM and is analogous to a disk storage on a typical desktop system.

3.1.3 Palm OS Network Library

There are two different Palm OS libraries to provide network services to applications: a net library and an Internet library.

The net library provides basic network services using TCP/IP via a socket API. It consists a netlib interface and a net protocol stack. The net library's API maps almost directly to the Berkeley UNIX socket API [36], except for some functions such as open, close, and set up the net library. An application using the Berkeley sockets could be compiled for the Palm OS with slight changes to the source code.

The net protocol stack runs as a subsystem of the operating system. The TCP/IP protocols are part of the stack. One or more network interfaces run inside the net protocol stack, such as point to point protocol (PPP) interface and serial link protocol (SLIP) interface.

Because devices using Palm OS have very limited amounts of memory, network support is an optional part of Palm OS. For this reason, the net library is loaded at runtime.

The Internet library provides Palm OS applications with easy access to World Wide Web documents. The Internet library uses the net library for basic network access and builds on top of the net library's socket concept to provide a socket-like API to the higher level Internet protocols like HTTP.

3.1.4 Communication Connections and Services

PC connectivity is an important part of Palm OS communication. A cradle is a device for connecting a Palm OS handheld to a PC. With a cradle and a software object called a conduit, a handheld application can exchange and synchronize data with an application running on a PC.

Palm OS based PDAs use an RS-232 serial port to provide connectivity. For example, the serial port could connect to a cradle or a modem. The Palm OS serial communications software provides serial communications capabilities, including byte-level serial I/O, connection management, and modem dialing capabilities.

Palm OS based PDAs also support infrared communications with other devices.

PDAs with certain software and hardware support wireless communications through a cell phone, which helps the handhelds to establish Internet connections in cellular networks.

Bluetooth wireless technology [35] is a standard and specification for small form-factor, low-cost, short range radio links between mobile PCs, mobile phones and other

portable devices. PDAs use Bluetooth technology to communicate wirelessly to other devices with Bluetooth support [1]. Palm OS Bluetooth support provides Bluetooth connectivity to Palm OS features, such as HotSync operations and PPP network access. In addition, there is an API that allows the development of Bluetooth-enabled applications [18].

3.1.5 Programming on Palm OS

Programming on Palm OS is different from programming for desktop applications. Palm OS applications are mostly single-threaded, event-driven programs. Only one program runs at a time.

Each application has a function `PilotMain()` that is equivalent to the function `main()` in C programs. To launch an application, the system calls function `PilotMain()` and passes it a launch code. The launch code may specify that the application is to become active and display its user interface (called a normal launch) or that the application should simply perform a small task and exit without displaying its user interface.

Palm OS is an event-based operating system, thus Palm OS applications contain an event loop. This event loop only starts in response to a normal launch. An application may perform work outside the event loop in response to other launch codes.

Most Palm OS applications contain a user interface made up of forms, which are analogous to windows in a desktop application.

In short, the Palm OS is intended for running small personal programs.

3.2 P-kernel

Network software is structured into layers which deal with different networking functions, such as transportation, inter networking, and link connection. Each layer contains a certain number of protocols. There are two main ways to implement network software. One way considers the network software as a sub-system of the kernel of an OS, for example, the TCP/IP stack of the Linux kernel [34]. This approach might be efficient but the context switches between user space and kernel space are required to pass messages from and to applications. Another approach considers implementation as a user process. For example, x-kernel is a framework to implement network software as a part of an application [16]. This approach is suitable for network software development on a non open OS platform or network simulations, making abstractions of details of an OS.

As the net library of the Palm OS is not open and the Palm OS does not support multitasking, a protocol development framework is needed to experiment with new protocols. P-kernel is a framework for implementing mobile and wireless network protocols on PDAs [33]. It is inspired by the x-kernel. The actualization of the framework is a communication application running on the Palm OS.

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

The p-kernel framework provides a set of data structures, subroutines, and slots. The framework is configurable with a protocol graph which specifies a set of protocols being implemented. Developers can add their own protocol implementation as data structures and subroutines to attain an actualization of the framework.

The main classes of objects in p-kernel are sessions, protocols, messages and events.

Session objects represent the local end-points of a channel. They are created dynamically when channels are opened, and are killed when the channels are closed. A session maintains the state of a channel end-point, e.g., values of sequence numbers. Sessions are used to send and receive messages through the channels.

Protocol objects are containers of sessions. A protocol object is a static entity, created when a protocol graph is instantiated, modeling a given protocol such as Mobile IP. Protocol objects open channels, resulting in the creation of session objects.

Messages are created dynamically and flow from one protocol or session to another. A message is normally made of a header part and a user data part. There are two message queues in p-kernel: an incoming message queue and an outgoing message queue. Messages in the incoming queue are put there by the low level protocols. Messages in the outgoing queue are created by applications or a specific protocol. To reduce the possibility of losing incoming messages, the incoming messages are processed prior to the outgoing messages.

The message-per-process model is implemented in the p-kernel. Each message is taken charge of by a thread when either it is received at the hardware level or it is created at the application level. A thread calls the procedures successively, which are used to implement tasks of the protocols and sessions which deal with message transmission. Therefore a thread handles a message from the beginning to the end of its existence. Typically, when a message flows down, the headers are attached to the message in every layer it passes. When a message flows up, the headers are stripped at every layer it goes through. A message goes through all the procedures without being copied.

One important feature of the message implementation in p-kernel is that there is only one buffer per message. Therefore the user has to allocate a large enough buffer to hold all the headers of the message.

Event is a common entry in protocol implementations. It is necessary to schedule events that happen after periods of time. For example, following the transmission of message, if an acknowledgement has not been received after a period of time, it

should be retransmitted. In this example, the event is a timeout. Moreover, the event concept is used to perform periodic functions, such as sending RA messages by routers.

The queue of events maintained by the p-kernel is orthogonal to any event structure maintained by the Palm OS [33]. An event includes the time remaining before the event fires, a function which is executed when the event is triggered, and arguments of the function. Processing of events in the p-kernel is triggered by the Palm OS timeout (a type of Palm OS event). An event is put in the event queue with a timeout. When the timeout occurs, the application event loop processes the event at the head of the event queue.

Threads are not implemented in the Palm OS. The p-kernel simulates multi-threads by using an event queue, incoming (pop) message queue, and outgoing (push) message queue.

X-kernel introduces a notion of Uniform Protocol Interface (UPI). The communication primitives supported by protocols and sessions are uniform and generic across all the layers. They are actualized by the protocol developer, using a function pointer mechanism. The UPI helps to achieve a uniform structure from one layer to another that considerably clarifies the protocol stack.

The framework is configurable with a protocol graph. Developers are allowed to put new protocols in the protocol graph, which is represented by an array as follows:

```
Prot1_init_array[] =
{
    {&KISS_prot1_init},
    {&IPv6_prot1_init},
    {&KISSTester_prot1_init}
};
```

This protocol graph includes three layer protocols: KISS, IP6 and KISSTester. Function `KISS_prot1_init`, `IPv6_prot1_init`, and `KISSTester_prot1_init`, create

a KISS protocol object, an IPv6 protocol object, and a KISSTester protocol object, respectively.

As p-kernel is a Palm OS application, it provides an entry point (function `PilotMain()`), initialization of components like main forms and protocols (function `App_Init()`), a control loop (function `App_Event_Loop()`), and termination (function `App_Finish()`). The function `PilotMain()` is defined as follows:

```
DWord PilotMain(Word wCmd, Ptr pInfo, Word fuLaunch)
{
    error = App_Init(protl_init_array, NUM_PROTOCOLS);
    App_Event_Loop();
    App_Finish();
}
```

When the entry function `PilotMain()` is entered, function `App_Init()` is called, the protocol graph is instantiated, the p-kernel OS is created, and every protocol of the graph is initialized. The dependencies within a protocol graph are defined programmable.

As mentioned in Section 3.1.5, a typical application on the Palm OS is driven by an 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 OS>
    <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 the usual event handling model of a Palm OS application.

Though p-kernel is built on top of Palm OS, the code is kept as abstract as possible. The developer only needs to be concerned with the p-kernel APIs and some Palm OS specific issues. That is, protocol implementation in P-kernel is mostly independent of the Palm OS. In fact, p-kernel makes it easy for developers to implement new protocol architectures because the details of the operating system are hidden in the framework. This feature makes the protocol implementation in p-kernel easy to port to other wireless device OSs. The p-kernel is written in C++.

Chapter 4

Design of Mobile Nodes

The goal of this thesis is to investigate a subset of Mobile IPv6 implementation on the Palm OS. The design of the mobility support functions of an MN in an IPv6 network is discussed in this chapter. As with the net library of the Palm OS it is not feasible to develop Mobile IPv6, a protocol development framework called p-kernel is used.

4.1 Motivation

PDAs are widely used as they are easily carried and allow users get information conveniently. Wireless communication is a valuable feature for PDAs.

Palm OS is a main operating system for PDAs. Therefore to enrich the wireless communications functionality of Palm OS, doing research on mobile and wireless protocols on the Palm OS is relevant. That is, experimenting with new mobile and wireless protocols at the network, data link, and physical layers.

Mobile IP is a mobility support protocol standard. It supports mobility in the network layer and is transparent to upper layers and application software. IPv6 is the next generation inter networking protocol that has enough address space for several billion wireless devices compared to IPv4. Moreover, IPv6 has built-in mobility

support. There is a movement in the wireless industry toward IPv6. For example, 3GPP has adopted IPv6 for their next generation of wireless network specification.

The Palm OS net library only supports a fixed set of protocols and is not open source. Thus, it is impossible to integrate new protocols into it. Since the net library is not usable as a vehicle for experimenting with the new mobile and wireless protocols, the Mobile IPv6 development in this thesis is based on a new network protocol development framework called p-kernel.

4.2 Software Structure

Figure 4.1: Software Structure

The p-kernel is a framework for protocol development. It runs as a module of a Palm OS application. It provides a programmable protocol graph allowing developers to put new protocols. IPv6 is implemented as a layer in the protocol graph and provides mobility support. There are two other layers: KISSTester layer and KISS layer. The KISSTester layer is used to test the sending and receiving of text which is displayed on the screen of a Palm handheld. The KISS layer implements the AX.25

protocol, and sends and receives packets in KISS mode.

When the application is entered, the protocol graph is instantiated and the IPv6 layer is initialized. The mobility support functions, such as movement detection and care-of address registration, are implemented in this layer. The IPv6 layer pushes messages down to the lower layer (KISS layer) to send messages, and pops received messages to the upper layer (KISSTester layer). This software structure is shown in Figure 4.1.

4.3 Software Scope

The experiment of Mobile IPv6 implementation is according to the Internet draft of Mobility Support in IPv6. Since the net library of the Palm OS can not be used to develop Mobile IPv6, the stack of Mobile IPv6 is built from scratch. Thus an incremental approach is adopted to cope with the complexity. In this thesis, we focus on the basic functions of Mobile IPv6. The movement detection and registration allow an MN to detect its movement and inform CNs and its HA. Therefore the MN can maintain its connectivity with other nodes all the time. These basic functions of the Mobile IPv6 are implemented under the following assumptions:

- Each MN has one permanent home address and home agent;
- Each MN attaches to one link at a time, therefore an MN only maintains a default router list with the same prefix, and its primary care-of address;
- All these addresses are reachable within their lifetime, so there is no need to use NUD messages to check the reachability of routers and the HA;
- All these addresses are unique on the link to which they are attached, so it is not necessary to use Neighbor Discovery messages to check the uniqueness of the link-local address when it is formed.

With limited power and limited resources, a Palm handheld is designed for users needing information quickly. It is necessary to reduce the amount of control messages sent. The policy for an MN is to limit the number of retransmissions to a relatively low number.

The IP security support is not implemented in this experiment.

4.4 Scenario Descriptions

Use case models are used to illustrate the scenarios in this implementation.

Use case models are used to capture the behaviors of a system from a user's point of view. The use case model includes actors and use cases. An actor represents a person or a thing that interacts with the system. Use cases represent what an actor wants to do with the system. Actors trigger use cases. A use case may include the behavior described by another use case. As show in the Figure 4.2, actors are represented by little stick people. Use cases are represented by ellipses. An arrow between two use cases represents that the source use case uses the behavior in the target use case.

4.4.1 Movement Detection

Figure 4.2 pictures the use cases of the mobility support system. An MN is the only actor in this system. When an MN boots, it is initialized according to configuration files.

Sending RS and **Receiving RA** are the main use cases to detect movements. An MN uses RS messages and RA messages to detect its movements. An MN may send RS messages or may wait for unsolicited (periodic) RA messages. In the use case **Sending RS**, an MN sends RS messages actively.

In the use case **Receiving RA**, an MN discovers new routers and on-link subnet

prefixes from RA messages. When an MN receives a RA message, it modifies its default router list that records the current on-link routers' address, prefix information, and corresponding care-of address based on this prefix. If the message includes a new prefix, it means that the MN has entered a new link. The MN will form a new care-of address according to the new prefix information. When the MN detects that it has moved to a new location, the use case **Registration** is called to notify its HA and CNs of the new care-of address.

Router entries have lifetimes. The use case **Checking Router Lifetime** is used to deal with each entry's lifetime. A timeout event is used to check the lifetime. When a router entry's lifetime expires, it is deleted from the default router list. 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 has moved out the network. The use case **Registration** is called for the node to de-register the care-of address with its HA and notify its CNs.

4.4.2 Binding Management

An MN maintains a binding update list of the nodes with which it is currently communicating, such as an HA and CNs. These behaviors are pictured in Figure 4.2.

The use case **Registration** uses use case **Sending BU** to send packets, which contain a Binding Update option, to the nodes in the MN's binding update list. To make sure its HA receives this packet, the MN asks the HA to reply with a packet carrying a Binding Acknowledgement. The MN will continue re-transmitting the packet with this Binding Update option to its HA until it receives a packet with a matching Binding Acknowledgement option or it moves to a new link.

Use case **Receiving BA** corresponds to the behaviors when a packet carrying a Binding Acknowledgement is received. If a binding is received, the MN stops sending Binding Updates. Otherwise, the binding entry is deleted from the binding update list.

Figure 4.2: MN's Use Case Diagram

Before the entry of an MN in a CN's binding cache expires, the CN can refresh the binding by sending a packet with a Binding Request option towards the MN. Use case **Receiving BR** includes the actions when an MN receives a packet including a Binding Request option. That is, the MN replies to the sender with a packet including a Binding Update option.

This mechanism makes an MN's up-to-date care-of address is known by its HA and CNs. Therefore an HA is capable of intercepting the packets addressed to an MN's home address and tunnel them to the MN. Also, the CNs that tend to send packets to the MN will use the MN's care-of address as the Destination Address in the IPv6 header.

The use case **Checking Binding Lifetime** checks the lifetime of the binding entries in the binding update list. A timeout event associates with the lifetime of each binding entry.

4.4.3 Address Autoconfiguration

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. The address is formed by combining a 64-bit address prefix with a 64-bit interface identifier [10] (Figure 4.3).

An MN has a link-local address, a global home address and a global care-of address. The prefix of the link-local address and the home address is fixed. The prefix of the care-of address is discovered by the received RA messages.

4.5 Data Structure

In this Mobile IPv6 implementation, the main data structures include `router_t`, `binding_update_list_t`, and `home_net_t`. They are described in more detail hereafter.

Figure 4.3: IPv6 Unicast Address

router_t

Structure `router_t` refers to the information of the routers found on the link. It is as follows:

```
struct router_t
{
    struct in6_addr router_addr;
    struct in6_addr prefix;
    UInt8  prefix_len;
    struct in6_addr care_addr;
    UInt32 life_time;
    Event  *router_timeout_event;
    struct router_t *prev;
    struct router_t *next;
}
```

This structure contains the router's address (`router_addr`), the link prefix (`prefix`) and its length (`prefix_len`), the care-of address (`care_addr`) which is based on this router, and the router entry's lifetime (`life_time`). There is an event called `router_timeout_event` which occurs when the router entry's lifetime expires. The pointers (`prev`, `next`) point to the previous and next node in this structure, respectively.

binding_update_list_t

An MN maintains a binding update list to record information for each Binding Update sent by the MN. Each binding entry in the list should have a valid lifetime. The structure is shown as follows.

```

struct binding_update_list_t
{
    UInt8 status;
    struct in6_addr to_addr;
    struct in6_addr care_addr;
    Event *update_timeout_event;
    UInt8 update_times;
    UInt32 life_time;
    Event *binding_timeout_event;
    struct binding_update_t *prev;
    struct binding_update_t *next;
};

```

The structure stores all bindings sent by the MN. It has an entry to the MN's HA and several entries to its CNs. Each entry has two status (**status**): accepted or not. It also contains the IP address of any node (mobile or stationary) to which an MN has sent a Binding Update (**to_addr**). The MN's care-of address (**care_addr**) that is sent in the Binding Update is also listed in the structure.

When an MN registers its care-of address to an HA, it schedules a timeout event (**update_timeout_event**) for re-transmitting a packet with a Binding Update option. Binding Updates should be sent only a limited number of times. The field **update_times** refers to how many times that the Binding Updates have been sent.

Every binding has a lifetime (**life_time**). When a binding has expired, the binding timeout event (**binding_timeout_event**) is triggered.

The pointers (**prev** and **next**) point to the previous and next node in the structure, respectively.

home_net_t

Each MN has its permanent home network. The structure of **home_net_t** refers to the information on the home network. The structure is as follows:

```
struct home_net_t{
    struct in6_addr home_agent;
    struct in6_addr home_addr;
    struct in6_addr home_prefix;
    int    prefix_len;
    struct in6_addr link_local_addr;
}
```

There are home agent address (**home_agent**), home address (**home_addr**), home prefix (**home_prefix**) and prefix length (**prefix_len**). Also an MN has a link local address (**link_local_addr**) which is formed by combining the link local prefix with the MN's interface identifier.

4.6 Activity Diagram

Activity diagrams represent the behavior of an operation as a set of actions. In this section, the activity diagrams represent the internal behaviors of the scenarios we describe in Section 4.4.

Figure 4.4 displays the graphical representations of activities. A filled circle represents the activity of start. A circle that consists a filled circle inside represents the activity of end. An activity is represented by an rectangle with circular corner. Activities are linked by automatic transitions which are represented by arrows. When

Figure 4.4: Activity Diagram Symbols

an activity terminates, the transition is triggered and the next activity starts. The logical decision expression is expressed by a diamond. It labels multiple output transitions of an action with different guard conditions. Activity diagrams show parallel behaviors between activities by using horizontal synchronization bars. With the synchronization bar, there are two kinds of symbols: fork and join. The fork splits an incoming transition into several concurrent outgoing transitions that fire together. The join merges transitions from concurrent regions into a single outgoing transition. A sending signal is symbolized by a convex pentagon, while a receiving signal is represented by a concave pentagon.

Main Diagram

Figure 4.5 illustrates the main routine of our Mobile IPv6 implementation. When the program is entered, the MN is initialized (**Initialization**). RS messages are sent (**Sending RS**). Following the activity is four concurrent activities: **Receive Msg**, **Router Tout**, **Binding Tout**, and **BU Tout**.

An MN waits for incoming messages and handles them in the activity **Receiving Msg**.

As we have mentioned before in Section 4.4 and Section 4.5, the router addresses and the bindings in the binding update list have lifetime. The activity **Router Tout**

Figure 4.5: Main Diagram

and the activity **Binding Tout** represent the actions, which deal with the lifetime of the router addresses and the bindings, respectively.

An MN waits for the answer from its HA after it sends a Binding Update to its HA. If it does not receive the answer after a period of time, it sends the Binding Update again. The activity **Binding Tout** handles these actions.

Receive Msg Diagram

Figure 4.6 shows the activities of **Receive Msg**. While an MN receives incoming messages, it checks the type of the messages. Upon receipt of a RA message, the activity of **Process RA** is triggered. Upon receiving a packet carrying a Binding Acknowledgement, the activity of **Process BA** fires. Upon receiving a packet carrying a Binding Request, the activity of **Process BR** is triggered. If it receives neither ICMP messages nor packets with a Destination Option header, the activity of **Process Other Msg** occurs.

Figure 4.6: Receive Msg Diagram

Process RA Diagram

Figure 4.7 shows the processing of an incoming RA message. The MN validates the RA message according to the requirements, which include validity checks of the IPv6 header and the ICMP message [24]. The message which does not satisfy the requirements is discarded. Then the MN tries to find the prefix option in the RA message. Since a RA message without a prefix option cannot provide the link prefix information, it is discarded by the MN. Otherwise, the MN gets the router information such as the router address, the lifetime of the router address, the link prefix, etc. Then the activity of **Detect Movement** is triggered.

Figure 4.7: Process RA Diagram

Detect Movement Diagram

Figure 4.8 illustrates the activities of movement detection. First of all, the MN compares the link prefix in the received router information with that of the default router. If the received link prefix is not new, that means the MN has not moved out and stays on the old link. Then the MN compares the address of the new received router with that existing in its default router list. If the router address already exists, the MN refreshes its lifetime according to the received message, the old router timeout event is canceled, and a new router timeout event is inserted. If the received router address is new and attached to the old link, a new router entry is added in the default router list, the new router is set as the default router, a router timeout event is inserted in the event queue.

If the link prefix is new, it indicates that the MN has moved out. That is, either it has returned to its home link, or has attached to a new foreign link. In the former case, the MN de-registers its care-of address with its HA, including informing its CNs. In the latter case, the MN forms its new care-of address, adds the router entry in its default router list, sets the new router as its default router, inserts a router timeout event, and registers its new care-of address with its HA, including informing its CNs.

Register Diagram

Figure 4.9 illustrates the processing of registration . Firstly, the MN decides to register or de-register, which is determined by a parameter of the activity set by the caller. To de-register, the binding update timeout event (if scheduled) and the binding timeout event are canceled.

In both cases (registration and de-registration), a packet carrying a Binding Update is sent to every node (HA and CNs) in the binding update list.

Figure 4.8: Detect Movement Diagram

Figure 4.9: Register Diagram

Binding Update Diagram

Figure 4.10 shows the Binding Update diagram. If a Binding Update is used to send to a HA, the Acknowledgement bit in the Binding Update option is set. A binding update timeout event is scheduled to make sure the HA replies a packet with a Binding Acknowledgement option.

Before sending a packet with a Binding Update option, the MN's care-of address is used as the Source Address in the packet's IPv6 header. In addition, the Home Address option is added in the Destination Option header. Then the packet is sent.

Figure 4.10: Binding Update Diagram

Process BA Diagram

Figure 4.11 shows the actions when receiving a packet carrying a Binding Acknowledgement. First of all, the MN checks the source address of the sender. If the address is already in the binding update list, which means that the received message is responded to a Binding Update sent before, then the MN validates the message.

After the validation, the status field in the Binding Acknowledgement option is checked. If it indicates that the Binding Update is accepted, then the MN updates the corresponding binding entry in its binding update list, indicating that the Binding Update has been acknowledged. As a result, the MN stops retransmitting the Binding Update. The binding update event is canceled. The lifetime of the binding is replaced by the lifetime in the Binding Acknowledgement option. Then the old binding timeout event is canceled, and a new binding timeout event is inserted.

If the status field in the Binding Acknowledgement option indicates that a Binding Update sent before is not accepted, the MN stops retransmitting the Binding Update. The binding update timeout event is canceled, and the binding is deleted.

Process BR Diagram

Figure 4.12 illustrates the actions when receiving a packet with a Binding Request option. At first, the MN checks the binding update list. If the sender is not in the list, the MN adds a new binding entry for the node. Then a packet with a Binding Update option is returned to the sender.

Router Tout Diagram

Figure 4.13 illustrates the behaviors when a router timeout event occurs. When the event is triggered, the router is deleted from the default router list.

If the default router list is empty, there are no valid routers for the MN. The MN is thus not reachable by the nodes in other subnets. Therefore the MN de-registers

Figure 4.11: Process BA Diagram

Figure 4.12: Process BR Diagram

its care-of address to its HA, including notifying its CNs.

If the default router list is not empty, and the default router is deleted because it has timed out, a router in the default router list is selected to be the default router.

BU Tout Diagram

Figure 4.14 shows the actions when a binding update timeout event fires. When the event occurs, it means that the valid acknowledgement of the sent Binding Update has not received during a period of time. Before retransmitting the Binding Update, the sending frequency is checked. If it is less than the default sending frequency, a Binding Update is retransmitted. Otherwise, the MN stops sending Binding Updates, and deletes the binding entry from the binding update list.

Figure 4.13: Router Tout Diagram

Figure 4.14: BU Timeout Diagram

Binding Tout Diagram

Figure 4.15 illustrates the actions when a binding timeout event is triggered. The MN cancels the binding timeout event and the binding update timeout event for the corresponding binding entry. In addition, the binding entry is deleted from the binding update list.

Figure 4.15: Binding Tout Diagram

Chapter 5

Implementation and Experiment

This chapter presents an implementation of Mobile IPv6 on the Palm OS. That is, a Mobile IPv6 stack implementation using a protocol development framework called p-kernel. This implementation is primarily concerned with the movement detection and care-of address registration of an MN.

This application is built on Palm OS 3.5. CodeWarrior for Palm Platform Release 6 [17] is used as a development tool.

5.1 Application Structure

A communication application on the Palm OS is built using the p-kernel. The application embeds a stack of protocols constructed within the p-kernel. From the point of view of the p-kernel, the application is a top level protocol called KISSTester protocol. Therefore the stack contains three layers: KISSTester, IPv6, and KISS layer. As Figure 5.1 shows, IPv6 layer is under the KISSTester layer and KISS layer is under the IPv6 layer. The implementation of the protocol stack is written in C.

A protocol graph represents a set of protocols implemented in the p-kernel. The definition of the graph for the stack of protocols is as follows:

Figure 5.1: Protocol Stack

```

/* number of protocols in the stack */
#define NUM_PROTOCOLS 3

/* array of protocols in the stack */
Prot1_init_array[] =
{
    {&KISS_prot1_init},
    {&IPv6_prot1_init},
    {&KISSTester_prot1_init}
};

```

The data member `NUM_PROTOCOLS` is defined as the number of protocols. In this experiment, the number of protocols is three.

An array called `Prot1_init_array` lists all the protocols embedded in the application, including the initialization functions of the protocols. The protocol initialization functions for KISSTester protocol, IPv6, and KISS protocol are `KISSTester_prot1_init()`, `IPv6_prot1_init()`, and `KISS_prot1_init()`, respectively.

When the application is entered, protocols in the protocol graph are initialized by calling function `App_Init()`:

```
App_Init(prot1_init_array, NUM_PROTOCOLS);
```

The function `App_Init()` calls the protocol initialization function of each protocol. The sessions contained in each protocol are opened by the protocol of their upper layer. Figure 5.2 shows the action of opening sessions. Using the function `IPv6_open()`, the protocol `KISSTester` opens IPv6 sessions. Invoking the function `KISS_open()`, the protocol `IPv6` opens KISS sessions. In this implementation, each protocol includes one session.

Figure 5.2: Open Sessions

5.1.1 Test Application

The test application, called KISS Tester, provides an interface to input text to be sent and display received text. The interface is illustrated in Figure 5.3. The screen is divided into two areas. Data contained in incoming packets is posted in the upper area. Data entered by the user in the lower area is sent in packets.

This application includes three layers. The flow of messages is illustrated in Figure 5.4 and Figure 5.5.

Figure 5.4 shows the message transmission flow. The `KISSTester` copies the text input by the user to a message, then pushes it down to the `IPv6` session by calling the function `IPv6_push()`. After appending an `IPv6` header to the message, the `IPv6` session pushes it down to the `KISS` sessions by invoking the function `KISS_push()`. The `KISS` session appends a `KISS` header to the message, and then converts it into

Figure 5.3: Application Interface

Figure 5.4: Message Transmission Flow

frames. The frames are passed over an RS-232 interface to a data radio.

Figure 5.5 illustrates the message reception flow. The incoming message is received by the KISS protocol through the RS-232 interface. Since there is only one KISS session in this implementation, the message is popped directly to the session. After the KISS header is stripped, the session calls the function `IPv6_Demux()` by which the message is passed up to the IPv6 layer. In this function, the IPv6 header information, which includes the information of the fixed header and the extension headers, is extracted from the message and handled. Then the message is popped to the IPv6 session. After the IPv6 header is stripped from the message, the message is passed up to the top layer KISSTester by the function `KISSTester_Demux()`. The data contained in the message is displayed in the upper area of the interface.

Figure 5.5: Message Reception Flow

5.1.2 IPv6 Layer

In the IPv6 layer, a subset of mobility support features are implemented: router solicitation, router advertisement, binding update, binding request, and binding acknowledgement. The source code is written in C.

Global Variables

The following global variables refer to the MN's main data structures to store information about the mobility support.

```
struct router_t *default_router_list, *default_router;  
struct binding_update_list_t *binding_update_list;  
struct home_net_t *home_net;
```

The variable `default_router_list` stores the information about routers discovered on the current link. The variable `default_router` refers to a default router.

The variable `binding_update_list` is used to store information about the nodes that the MN is currently communicating with, such as its HA and CNs.

Each MN has its permanent home network information. The variable `home_net` includes the MN's home address, care-of address, and link-local address. The fixed link-local address is the combination of the link-local prefix and the node's 64-bit interface identifier. In this case, the data radio's call sign is used to create an interface identifier with IEEE EUI-64 format [2].

Description of Main Functions

Following are the main functions for the mobility support implementation in this experiment.

- `router_solicitation()`

This function is used to send RS messages when an MN is booted.

- `ra_process()`

This function is called when receiving a valid RA message. The router information is stored by this function. The logical details are shown in the activity diagram of Process RA (Figure 4.7).

- `find_router()`

After obtaining the router information, this function is used to detect the movement of the MN and make the decision to register. It corresponds to the activity diagram of Detect Movement (Figure 4.8).

- `registration()`

This function is used to send Binding Updates to the MN's HA and CNs. It corresponds to the activity diagram of Register (Figure 4.9).

- `recv_bindack()`

Upon receiving a Binding Acknowledgement, this function is called to check whether or not a binding is accepted. It corresponds to the activity diagram of Process BA (Figure 4.11).

- `recv_bindreq()`

When receiving a Binding Request, this function responds by returning a Binding Update. It corresponds to the activity diagram of Process BR (Figure 4.12).

Implementation of the Main Routine

First of all, an event is scheduled to send RS messages. The source code is as follows:

```
/* set the time interval of sending Router Solicitation messages */
time = ROUTER_SOLICITATION_PERIOD;
/* schedule an event to send Router Solicitation messages */
evSchedule(os_ptr, router_solicitation, self, time);
```

The function `evSchedule()` has four parameters: `os_ptr`, `router_solicitation`, `self`, and `time`. The variable `os_ptr` refers to the p-kernel OS. The parameter `router_solicitation` refers to a function executed when the event fires. The parameter `self` refers to the IPv6 protocol. The parameter `time` defines the time interval between sending each consecutive RS message.

After its initialization, the MN waits for incoming messages. After receiving incoming messages, the KISS session calls the function `IPv6_Demux()` to pass messages up to the IPv6 layer (Figure 5.5). The function `IPv6_Demux()` extracts the data contained in the incoming messages and handles different kinds of messages.

The source code of function `IPv6_Demux` is as follows:

```

1. PkReturn IPv6_Demux
2.   (
3.     Pk_Protocol      *self,
4.     void /*Sessn */  *lls,
5.     void /*Msg_ptr*/ *data
6.   )
7. {
8.   Msg *msg = (Msg *)data;
9.   char *buf;
10.  struct in6_addr router_address;
11.  struct router_t *new_router;
12.  struct icmp6_hdr *icmpv6hdr;
13.  IPv6_STATE_STRUCT *state_ptr = (IPv6_STATE_STRUCT *)self->Get_State();

14.  /* get the CURRENT_HEAD of the msg */
15.  buf = (char *)pmsgPeek(msg,0);
16.  /* point to IPv6 header information in the receiving message */
17.  state_ptr->ipv6_hdr = (struct ipv6hdr *)buf;

18.  /* if next header is NEXTHDR_ICMP, handling the ICMPv6 message*/
19.  if ( state_ptr->ipv6_hdr->next_header == NEXTHDR_ICMP) {
20.    /* point to icmpv6 message */
21.    buf += IPv6_HEADER_LENGTH;
22.    icmpv6hdr = (struct icmpv6_hdr *)buf;
23.    switch (icmpv6hdr->icmpv6_type){

```

```
24.      /* if it is Router Advertisement messages, handling Router
25.      Advertisement messages */
26.      case ND_ROUTER_ADVERT:
27.          /* get the router information from the receiving Router
28.          Advertisement message */
29.          new_router = ra_process (buf, self);
30.          /* detect the movement of the MN */
31.          find_router(new_router, self);
32.          break;
33.      /* icmp other messages: discard */
34.      default:
35.          FrmAlert(DiscardICMPAlert);
36.          break;
37.  }
38.  }

39.  /* receive a packet with a Destination header */
40.  if ( state_ptr->ipv6_hdr->next_header == NEXTHDR_DEST) {
41.      /* point to destination option message */
42.      buf += IPv6_HEADER_LENGTH;
43.      /*skip the padding */
44.      for (int i=0; i<4; i++)
45.          if (buf[i] == 0)
46.              buf++;
47.      /* point to the option type */
48.      struct tlv_head_t *tlv_head = (struct tlv_head_t *)buf;
49.      switch (tlv_head->type){
50.          /* receive a packet with a Binding Acknowledgement option */
51.          case BINDACK_TYPE:
52.              /* handle the Binding Acknowledgement option */
53.              recv_bindack(buf, self);
```

```
54.             break;
55.             /* receive a packet with a Binding Request option */
56.             case BINDREQ_TYPE:
57.             /* handle the Binding Request option */
58.                 recv_bindreq(self);
59.                 break;
60.             default:
61.                 break;
62.         }
63.     }

64. /* pop a message to the IPv6 session */
65. Sessn *ipv6_sessn;
66. /* get IPv6 session */
67. ipv6_sessn = (Sessn *) (self->Get_Session());
68. if (!ipv6_sessn)
69. {
70.     FrmAlert(IPv6GetSessionAlert);
71.     return PK_FAILURE;
72. }
73. /* pop the data */
74. IPv6_Pop(ipv6_sessn,data);
75. return PK_FINISHED;
76. }
```

The parameters of the function `IPv6_Demux()` are shown from Line 3 to 5. The first parameter (`self`) is the protocol IPv6 itself. It includes the protocol state, a pointer to the p-kernel OS, and its opening sessions. The second parameter (`lls`) represents a low level session of IPv6. It is the session that calls the function `IPv6_Demux()`. The third parameter (`data`) points to the message received in the IPv6 layer.

The body of the function includes two switches. The first case of the first switch (Line 26) corresponds to the handling of RA messages. If an MN receives a RA message, the function `ra_process()` (Line 29) is called to process the message. This function returns a pointer to a variable (`new_router`) representing the received router information. The variable is taken as a parameter by another function called `find_router()` (Line 31). The function `find_router()` detects the MN's movement by processing the router information, and decides whether or not to register the MN's current location with other relative nodes.

The cases of the second switch (Line 49) correspond to receiving packets with a Destination header. If a packet with a Binding Acknowledgement option is received, the function `recv_bindack()` (Line 53) is called to check whether or not the binding is accepted. Upon receiving a packet carrying a Binding Request, the function `recv_bindreq()` (line 58) is called. A packet with a Binding Update option is returned to the sender.

After the processing of the two switches, the incoming messages are popped to the IPv6 session (Line 74) (Figure 5.5).

This implementation only handles incoming messages with an ICMPv6 header or a Destination Option header. Thus in the processing of the IPv6 packets, the extension header is assumed to be either the ICMPv6 header or the Destination Option header.

5.2 Testing Environment

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, the Simple IPv6 Transition (SIT) is used to tunnel IPv6 packets over the IPv4 routing infrastructure. This technique uses the embedded IPv4 address structure called IPv4-compatible IPv6 address (Figure 5.6), which eliminates the need for tunnel configuration. The Linux workstation sends IPv6 packets encapsulated in

IPv4. Figure 5.7 shows the exchange of a packet header from a source machine to a destination machine.

Figure 5.6: IPv4-Compatible IPv6 Address

Figure 5.7: IPv6 Encapsulated in IPv4

The testing hardware includes a Palm handheld, a Linux desktop, and data radios which are connected to computers via serial cables.

At the moment, the MN running on the Palm handheld can send out RS messages and receive RA messages from the Linux workstation that acts as a router and an HA.

A registration test between two palm handhelds is also conducted (Figure 5.9). The average duration for a palm handheld to receive a BA message after sending out a BU message is around 2 seconds, that is, $T_{receive} - T_{send} = 2$ seconds.

Figure 5.8: Test Environment

Figure 5.9: Registration Test

Chapter 6

Conclusion

6.1 Summary

This thesis is mainly about the design and implementation of Mobile IPv6 on a PDA. An experiment of the wireless communications between a PDA and a router has been conducted.

The third-generation wireless network technologies are the trend of the future. The wireless data network technologies such as GPRS enable packets to transfer between mobile stations and the Internet. Thus the third generation wireless data networks will be all IP-based networks.

One problem in the normal IP is the lack of mobility support. A node only has one address based on its home link after it is booted. When the node roams to another subnet, it cannot be reached by this address. Mobile IP is designed to solve the mobility problem by introducing a new address architecture. Every MN has two addresses: a home address and a care-of address. The home address is used to identify the MN itself. The care-of address refers to the new location when an MN is attached to a new link. With the two level addresses, an MN is able to change its point of attachment without losing reachability or breaking connections. Mobile

IP provides mobility support in the IP layer. It is transparent to the upper layers and applications. Thus the existing network infrastructure and applications are not required to be changed while implementing the Mobile IP. There are two versions of Mobile IP: Mobile IPv4 and Mobile IPv6. Mobile IPv4 is an extension of IPv4 with mobility support. Mobile IPv6 is the built-in mobility support of IPv6 which is the next generation Internet protocol. IPv6 is intended for solving some problems of IPv4, such as not enough address space and no built-in mobility support. There is a movement in the wireless industry towards IPv6.

With the enhanced features of IPv6, such as enough address space, Neighbor Discovery protocol, address autoconfiguration, built-in security support, and built-in mobility support, Mobile IPv6 provides an efficient mobility solution that includes enhanced mobility transparency, built-in smooth handoffs, and home agent dynamically discovery. Thus Mobile IPv6 is a solution to provide IP mobility support for the next generation Internet.

PDAs are widely used in mobile computing and Palm OS is the de facto operating system standard for PDAs. Wireless devices will support IP services in the third generation all IP-based wireless networks. Research on mobile and wireless protocols on the Palm OS is needed for wireless devices. In this thesis, an experiment of the Mobile IPv6 implementation is built on the Palm OS. As the net library of the Palm OS is only for IPv4, it could not be used in the implementation. Without the support of the net library on the Palm OS, a protocol development framework is needed for the implementation of protocols. P-kernel is a protocol development framework to provide a communication model on the Palm OS. It is configurable and keeps abstract of details of Palm OS programming. Thus the Mobile IPv6 implementation are done using the p-kernel.

The main contribution of this thesis is a partial implementation of the Mobile IPv6 on the Palm OS. The implementation of Mobile IPv6 is developed from scratch. To cope with the complexity of the implementation, an incremental approach is adopted. The experiment focuses on control aspects of Mobile IPv6. The following basic functions are implemented: router solicitation, router advertisement, binding update,

binding request, and binding acknowledgement.

The implementation on Palm OS brings challenges because of PDAs with low memory, low speed processor, and small screen. It is not feasible to run a heavy weight software on PDAs. To solve this resource limitation problem, a light weight protocol implementation framework – p-kernel has been used. In addition, the protocol is also simplified. A scope of this protocol implementation is defined, for example, only one primary care-of address is considered, and the home network is assumed to be stable. Thus not all the mobility support functions are implemented, such as home network renumbering. Only the basic mobility functions are supported, such as router discovery and care-of address registration functions. The retransmission times of messages are also minimized. Data structures are designed to reduce redundancy. Moreover, a simple memory management model for messages are applied.

6.2 Future Work

Data transfer is not supported in this 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 impossible to add Mobile IPv6 on 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 the near future, the implementation will be improved in several ways. Since the implementation only supports active open of a single session at the moment, features such as multiple sessions and passive open of sessions need to be handled. In addition, the Mobile IPv6 protocol elements supported by the implementation will be enriched. Control mechanisms, such as dynamic home agent address discovery, will be considered. The future work will also focus on data transfer mechanisms, such as reception of packets using IPv6 de-capsulation and transmission of packets with

the Home Address option. Furthermore, routing mechanisms, and security support of Mobile IPv6 such as authentication, will be implemented.

Appendix A

Publications

1. Zhao-shu Zeng and Michel Barbeau. An Experimental Implementation of Mobile IPv6 on a Personal Digital Assistant [42]. IEEE ICT2001 (International Conference on Telecommunications). Bucharest, Romania, June 2001.

Bibliography

- [1] Bluetooth Wireless Technology Solutions. <http://www.widcomm.com/products>, April 2001.
- [2] IEEE Guidelines for 64-bit Global Identifier (EUI64) Registration Authority. <http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>, January 2001.
- [3] W. A. Beech, D. E. Nielsen, and J. Taylor. AX.25 Amateur Packet-Radio Link-Layer Protocol Version 2.2. <http://www.tapr.org/tapr/html/Fax25.html>, November 1997.
- [4] B. Bettstetter, H. Vogel, and J. Eberspacher. GSM Phase 2+ General Packet Radio Service GPRS: Architecture, Protocols, and Air Interface. *IEEE Communications Surveys*, 2(3), Third Quarter 1999.
- [5] D. Black. Differentiated Services and Tunnels. The Internet Engineering Task Force, Network Working Group, RFC 2983, October 2000.
- [6] J. Bound, M. Carney, and C. Perkins. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). The Internet Engineering Task Force, Network Working Group, Internet Draft, April 2001.
- [7] A. T. Campbell, J. Gomez, S. Kim, A. G. Valko, C.-Y. Wan, and Z. R. Turanyi. Design, Implementation, and Evaluation of Cellular IP. *IEEE Personal Communications*, pages 42–49, August 2000.

- [8] M. Chepponis and P. Karn. The KISS TNC: A Simple Host-to-TNC Communications Protocol. In *Proceedings of the 6th Computer Networking Conference*, August 1987.
- [9] S. Deering and R. Hinden. Internet Protocol Version 6 (IPv6) Specification. The Internet Engineering Task Force, Network Working Group, RFC 2460, December 1998.
- [10] S. Deering and R. Hinden. IP Version 6 Addressing Architecture. The Internet Engineering Task Force, Network Working Group, Internet Draft, October 2000.
- [11] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing. The Internet Engineering Task Force, Network Working Group, RFC 2267, January 1998.
- [12] Mobile Wireless Internet Forum. <http://www.mwif.org>, 2001.
- [13] A. Fuggetta, G. P. Picco, and G. Vigna. Understanding Code Mobility. *IEEE Transactions on Software Engineering*, 24, 1998.
- [14] IP Telephony Working Group. Ip Telephony. <http://www.ietf.org/html.charters/iptel-charter.html>, 2001.
- [15] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. SIP: Session Initiation Protocol. The Internet Engineering Task Force, Network Working Group, RFC 2543, March 1999.
- [16] N. C. Hutchinson and L. L. Peterson. The x-kernel: An Architecture for Implementing Network Protocols. *IEEE Transaction on Software Engineering*, 17(1):64–76, January 1991.
- [17] Metrowerks Inc. CodeWarrior for Palm Platform Release 6.0. <http://www.codeWarrior.com>.
- [18] Palm Inc. Palm OS Bluetooth Support. http://www.palmos.com/dev/tech/bluetooth/palm_bluetooth_whitepaper.pdf, April 2001.

- [19] Palm Inc. Palm OS Platform. <http://www.palmos.com>, April 2001.
- [20] D. B. Johnson and C. Perkins. Mobility Support in IPv6. IETF Internet Draft, November 2000.
- [21] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. The Internet Engineering Task Force, Network Working Group, RFC 2401, November 1998.
- [22] SUN Microsystem. Solaris IPv6. <http://www.sun.com/solaris/ipv6>, April 2001.
- [23] T. Narten. Neighbor Discovery and Stateless Autoconfiguration in IPv6. *IEEE Internet Computing*, 3(4):54–63, July-August 1999.
- [24] T. Narten, E. Nordmark, and W. Simpson. Neighbor Discovery for IP Version 6 (IPv6). The Internet Engineering Task Force, Network Working Group, RFC 2461, December 1998.
- [25] T. Narten and S. Thomson. IPv6 Stateless Address Autoconfiguration. The Internet Engineering Task Force, Network Working Group, RFC 2462, December 1998.
- [26] National University of Singapore. Mobile IPv6 Implementation. <http://mip.ee.nus.edu.sg>, April 2001.
- [27] Helsinki University of Technology. MIPL Mobile IPv6 for Linux. <http://vesper.tky.hut.fi/mip>, April 2001.
- [28] C. Perkins. IP Mobility Support. The Internet Engineering Task Force, Network Working Group, RFC 2002, October 1996.
- [29] C. E. Perkins. *Mobile IP: Design Principles and Practices*. Addison-Wesley, Reading, Massachusetts, 1998.
- [30] Third-Generation Partnership Project. <http://www.3gpp.org>, 2001.
- [31] Wireless Application Protocol. <http://www.wapforum.org>, 2001.

- [32] Microsoft Research. Mobile IPv6 Implementation. <http://research.microsoft.com/msripv6/>, April 2001.
- [33] S. Robinson and M. Barbeau. The P-kernel: A Framework for Mobile and Wireless Network Protocol Implementation on Personal Digital Assistants. <http://scs.carleton.ca/~barbeau>, 2000.
- [34] S. T. Satchell and H. B. J. Clifford. *Linux IP Stacks Commentary*. CoriolisOpen Press, October 1999.
- [35] Bluetooth SIG. Specification of the Bluetooth System. http://www.bluetooth.com/developer/specification/Bluetooth_11_Profiles_Book.pdf, February 2001.
- [36] W. R. Stevens. *UNIX Network Programming*. Prentice Hall PTR, Upper Saddle River, NJ, 1998.
- [37] A. S. Tanenbaum. *Computer Networks*. Prentice Hall, Englewood Cliffs, NJ, 1988.
- [38] Tucson Amateur Packet Radio (TAPR). Introduction to Packet Radio. <http://www.tapr.org/tapr/html/pktf.html>, January 1999.
- [39] Carnegie Mellon University. The Monarch Project: Protocols for Adaptive Mobile and Wireless Networking. <http://www.monarch.cs.cmu.edu>, April 2000.
- [40] A. G. Valko. Cellular IP: a New Approach to Internet Host Mobility. *ACM Computer Communication Review*, January 1999.
- [41] I. Wade. *NOSintro: TCP/IP over Packet Radio*. Dowermain Ltd, Maxet House, Liverpool Road, Luton, Bedfordshire, United Kingdom, 1992.
- [42] Z. Zeng and M. Barbeau. An Experimental Implementation of Mobile IPv6 on a Personal Digital Assistant. <http://scs.carleton.ca/~barbeau>, 2001.