Project Number:	**IST-2001-32603**
Project Title:	**6NET**
CEC Deliverable Number:	**32603/ULANC/DS/4.1.4/A1**
Contractual Date of Delivery to the CEC:	December 31st 2004
Actual Date of Delivery to the CEC:	February 8th 2005
Title of Deliverable:	Final MIPv6 Support Guide
Work package contributing to Deliverable:	WP4
Type of Deliverable*:	R
Deliverable Security Class**:	PU
Editor:	Martin Dunmore
Contributors:	Martin Dunmore, Reinhard Ruppelt, Tommi Saarinen
Reviewers:	Chris Edwards

* Type: P - Prototype, R - Report, D - Demonstrator, O - Other

** Security Class: PU- Public, PP – Restricted to other programme participants (including the Commission), RE – Restricted to a group defined by the consortium (including the Commission), CO – Confidential, only for members of the consortium (including the Commission)

Abstract:

This document provides a guide for people wishing to deploy a Mobile IPv6 (MIPv6) testbed at their site. We describe what infrastructure is required to support MIPv6 and detail the installation, configuration and operation of the most suitable implementations. Three case studies of MIPv6 testbeds within 6NET are provided.

Keywords:

mobile, IPv6, MIPv6, guide, site, implementation, host, router, mipl, kame, xp, ios

Executive Summary

This deliverable provides a guide for people wishing to deploy a Mobile IPv6 (MIPv6) testbed at their site. Although the MIPv6 protocol reached RFC status in 2004, it is still very much an immature protocol and further improvements and extensions are ongoing with the IETF mip6 working group.

Although MIPv6 implementations have been around since 1998 most have fallen out of date as the MIPv6 protocol has progressed over the years. However, there exists a handful of available implementations that are fairly up to date with either MIPv6 draft version 24 or RFC 3775 (which is based on draft version 24). Yet these implementations can still differ slightly in their supported features and are not likely to be completely 100% interoperable in most cases.

Whilst reading this document, it is important to understand that this is a snapshot of available MIPv6 implementations. Information herein may soon become outdated as the MIPv6 protocol develops and new, improved implementations become available.

In this deliverable we describe the Mobile IPv6 protocol and identify the likely implementations that people can gain access to and try out for themselves. We go on to provide basic HOW-TOs for MIPv6 implementations for Linux, BSD, Cisco and Microsoft Windows systems and then present 3 case studies of 6NET partners that have deployed MIPv6 testbeds at their sites.

Table of Contents

1 Introduction

This document provides a guide for people wishing to deploy a Mobile IPv6 (MIPv6) testbed at their site. Although the MIPv6 protocol reached RFC status in 2004, it is still very much an immature protocol and further improvements and extensions are ongoing with the IETF mip6 working group.

Although MIPv6 implementations have been around since 1998 (beginning with Lancaster University's implementation for Linux) most have fallen out of date as the MIPv6 protocol has progressed over the years (including the Lancaster University implementation). However, there exists a handful of available implementations that are fairly up to date with either MIPv6 draft version 24 or RFC 3775 (which is based on draft version 24). Yet these implementations can still differ slightly in their supported features and are not likely to be completely 100% interoperable in most cases.

There are many people within the Internet community who wish to deploy and gain experience of MIPv6 in order to help realise the new mobile applications and services of the next generation Internet. This document describes what infrastructure is required to deploy a general MIPv6 testbed and details the installation, configuration and operation of the most suitable implementations currently available.

Whilst reading this document, it is important to understand that this is a snapshot of available MIPv6 implementations. Information herein may soon become outdated as the MIPv6 protocol develops and new, improved implementations become available.

The rest of this document is structured as follows. The following next section provides an overview of the Mobile IPv6 protocol. Section 3 lists the available MIPv6 implementations know to date. Section 4 describes some deployment considerations. Sections 5, 6, 7 and 8 detail the installation, configuration and usage of MIPv6 implementations from Cisco, MIPL, KAME and Microsoft respectively. Finally, section 9 describes three case studies of 6NET partners that have deployed MIPv6 testbeds at their sites, namely Fraunhofer Fokus, the University of Oulu and Lancaster University.

2 Mobile IPv6 Overview

The Mobile IPv6 (MIPv6) protocol [1] is a proposed standard by the IETF to provide transparent host mobility within IPv6. The protocol enables a Mobile Node to move from one network to another without the need to change its IPv6 address. A Mobile Node is always addressable by its *home address*, which is the IPv6 address that is assigned to the node within its home network. When a Mobile Node is away from its home network, packets can still be routed to it using the node's home address. In this way, the movement of a node between networks is completely invisible to transport and other higher-layer protocols.

Mobile Nodes participating in the MIPv6 protocol each have a persistent *home address*, which can be used to address the Mobile Node irrespective of its current point of attachment to the IPv6 network. The IPv6 network which matches the home address' prefix is known as the *home network*. Mobile Nodes also adopt a *Home Agent* - an IPv6 capable router directly connected to the home network. This process may either be static, or dynamic, via the MIPv6 Home Agent discovery mechanism. The Home Agent is responsible for the interception and forwarding of IPv6 packets to the Mobile Node which are incorrectly routed to the home network while the Mobile Node is away from home.

When a Mobile Node is attached to its home network it operates as any other network node, so no special routing is required. When a Mobile Node moves to a foreign network, it uses IPv6 autoconfiguration to discover the new network and to allocate a care-of address within the address space of that network. However, to ensure that IPv6 packets destined for the Mobile Node's home address reach the proper location as efficiently as possible, the routing information pertaining to the Mobile Node's home address must be updated in both the Home Agent and any relevant Correspondent Nodes. MIPv6 provides this functionality by the introduction of a bindings cache on the Mobile and Correspondent Nodes and binding update messages which are transmitted in a new IPv6 extension header called the mobility header.

2.1 Bindings Cache

The relationship between a Mobile Node's home address and its current care-of address is known as a *binding*. All Nodes participating in MIPv6 are required to maintain a table of these bindings in a *binding cache*. One entry is held in the binding cache for each Mobile Node with which communication is currently taking place. The binding cache holds four pieces of information per binding which are central to the operation of MIPv6, as illustrated by Table 1 (other fields are present to ensure correct ordering of control messages, but these are omitted for clarity). The home address forms the key field of the cache.

Home Address	Care of Address	Lifetime	Home Agent
3ffe:2101:0:b00::10	2001:2101:0:a00:260:97ff:fe8b:4c56	120	Yes
3ffe:2101:0:b00::15	2001:2101:0:b00:a00:6aff:fe2b:137c	43	No

Table 1 Mobile IPv6 Bindings Cache

When a node wishes to transmit an IPv6 packet to a remote host, the home address field of the binding cache is searched for the IPv6 address of that host. If no match is found, the packet is transmitted according to the normal IPv6 routing tables. However, if a match is found, then the packet is encapsulated prior to transmission, to redirect the packet to the care-of address specified in the binding cache. This ensures optimal routing to the Mobile Node's current location. The form this encapsulation takes is dependant on the state of the 'Home Agent' flag stored in the binding cache entry.

2.2 Home Agent Operation

If the 'Home Agent' flag is set in a binding cache entry, then the node maintaining that cache is acting as a Home Agent for the Mobile Node. If this is the case, and the packet was entered through a forwarding context, then the packet is encapsulated using IPv6 in IPv6 tunnelling, as illustrated in Table 2.

IPv6 Header(Outer)	IPv6 Header (Inner)	Transport Header	Payload
Source Address: Home Agent	Source Address: Correspondent Node	TCP/UDP	Data
Dest. Address: Care of Address	Dest. Address: Home Address		
← 40 bytes →	← 40 bytes →		

Table 2 IPv6 in IPv6 Encapsulation

IPv6 tunnelling is used by Home Agents to forward IPv6 packets misrouted to a Mobile Node's home network while it is away from home, as illustrated in Figure 1. Tunnelling has the advantage of preserving the complete original IPv6 packet, which is important as any modification to an IPv6 header could cause problems with higher layer protocols, such as TCP. Intercepted packets are tunnelled directly to a Mobile Node's current care-of address, where they are subsequently decapsulated by the Mobile host.

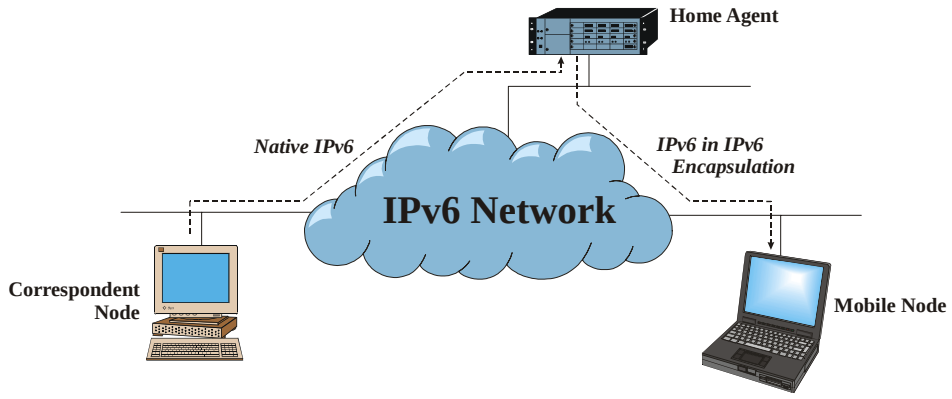


Figure 1 MIPv6 Routing to Mobile Nodes (Pre Route Optimisation)

2.3 Correspondent Node Operation

If the 'Home Agent' flag is cleared in a binding cache entry, or the packet was not received from a forwarding context, then an IPv6 routing header is used to redirect the IPv6 packet through the relevant care-of address, as shown in Figure 2.

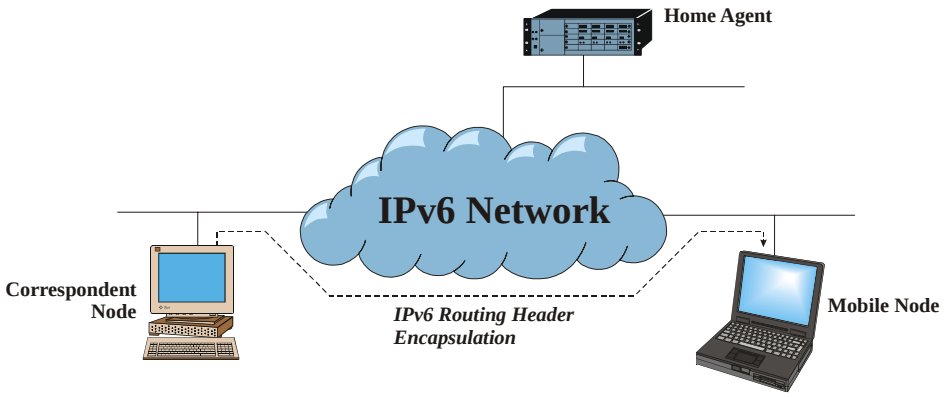


Figure 2 - Mobile IPv6 Routing to Mobile Nodes (Post Route Optimisation)

IPv6 Header	IPv6 Routing Header	Transport Header	Payload
Source Address: Correspondent Node Dest. Address: Care of Address	Next Hop Address: Home Address	TCP/UDP	Data
← 40 bytes →	← 24 bytes →		

Table 3 IPv6 Routing Header Encapsulation

As can be seen from Table 3, the use of the IPv6 routing header reduces the effective bandwidth required for the encapsulation of the packet in comparison to IPv6 in IPv6 tunnelling by 16 bytes. This reduction in packet size is possible due to spatial redundancy - i.e. if IPv6 tunnelling were used then the IPv6 source address of both the inner and outer IPv6 headers would be identical, resulting in a waste of bandwidth.

2.4 Binding Cache Coherence

The use of the binding cache and IPv6 encapsulation provides a mechanism to enable optimal routing to Mobile hosts. This mechanism, however, relies on the bindings contained within that cache being accurate and up to date. Indeed, to protect against total machine failure (which is common in a mobile environment due to battery life constraints, etc.) and long periods of network disconnection by Mobile Nodes, binding cache entries for a Mobile Node must persist even after a period of total disconnection or loss of state by Mobile or Correspondent Nodes.

Mobile IPv6 maintains binding cache coherence through the use of binding update, binding acknowledgement and binding request messages. The remainder of this section describes these messages in detail, and how they interoperate to provide accurate and timely binding cache coherence.

Binding update, acknowledgement and request messages are all carried inside IPv6 destination options, each with their own destination option type. Utilising IPv6 destination options gives several advantages over less integrated methods of control messaging. Firstly, messages can be placed inline with the header of existing IPv6 packets, thereby reducing the packet transmission overhead of the message. Secondly, as no transport layer protocol is involved in the transmission of the message, fewer issues exist concerning the blocking of control messages by firewalls.

Binding Update Messages

Binding updates are transmitted by Mobile Nodes to Home Agents and Correspondent Nodes to create or update the entry in their binding cache relating to that Mobile Node's home address.

Binding updates can be generated at any time by Mobile Nodes, but are always transmitted upon the detection of an IPv6 packet which has travelled via an IPv6 in IPv6 tunnel from that node's Home Agent. The reception of such a packet indicates that the Correspondent Node that generated the packet currently has no binding for this Mobile Node (else the packet would have been delivered via an IPv6 routing header). In order to guarantee that 'stale' bindings are not indefinitely maintained by binding caches, Mobile IPv6 employs a soft state mechanism to purge out of date bindings. Every binding update contains a lifetime field, which specifies, in seconds, how long the binding is valid for. After this lifetime expires, the binding is removed from the binding cache. The lifetime value is set and refreshed by the corresponding lifetime field contained within binding update messages. A lifetime value of zero in a binding update indicates removal of the relevant binding.

Binding Acknowledgement Messages

Unlike binding updates, binding acknowledgements are sent to Mobile Nodes by Correspondent Node and Home Agents. They provide control feedback to Mobile Nodes in response to binding updates, and are used to provide reliable binding update delivery and to indicate any errors which are generated during the remote processing of binding updates. Mobile Nodes match binding acknowledgements with their corresponding binding updates by the comparison of sequence numbers.

Binding Request Messages

Correspondent Nodes and Home Agents detecting an entry in their binding cache which is nearing expiry may decide to send a binding request message to the respective Mobile Node. The receipt of a binding request message by a Mobile Node results in the transmission of a new binding update to the source of that binding request. This mechanism enables Correspondent Nodes to avoid short periods of sub-optimal routing, due to the expiry of an accurate binding.

Binding Update List

As a Mobile Node roams from network to network, it is essential that binding update messages are transmitted to that node's Home Agent and Correspondent Nodes as soon as possible, in order to facilitate a fast handoff. Mobile Nodes therefore cannot rely on the soft state timeout mechanism used in binding caches to refresh stale bindings maintained by Correspondent Nodes (typical binding lifetimes are of the order of minutes). An additional data structure, the *binding update list*, is therefore kept by Mobile Nodes, which maintains state on any Correspondent Nodes or Home Agents. Figure 3 illustrates the binding update list.

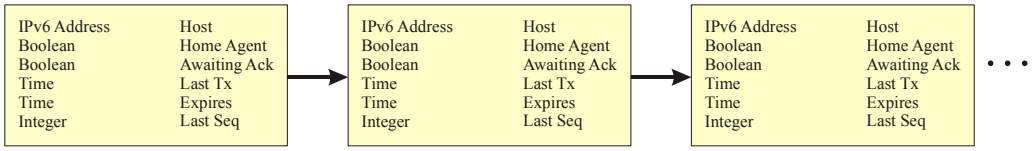


Figure 3 Mobile IPv6 Binding Update List

The binding update list contains one entry for every Correspondent Node or Home Agent to which a binding update has been sent. List entries contain information such as the address and time at which the binding update was transmitted, the state of any unacknowledged updates, the lifetime of the binding, a Home Agent flag, and the sequence number of the last transmission. Binding list entries are garbage collected from the binding update list as the respective binding expires.

The maintenance of the binding update list allows for significantly faster handoff performance. After a handoff has been detected and autoconfiguration has been completed, the binding update list is traversed, and a binding update message transmitted to every node contained within the list, thereby updating the binding caches of any active Correspondent Nodes.

2.5 Proxy Neighbour Discovery

Since packets destined for a Mobile Node may be incorrectly routed to its home network, placing Home Agents within an IPv6 edge router would allow the efficient interception of these packets, as they would likely travel through that router. However, the assumption that the packets will automatically reach this edge router cannot be relied upon. For example, consider the case of a Correspondent Node located on a Mobile Node's home network. If the Correspondent Node were to send a packet to that Mobile Node, its routing table would dictate that the Mobile Node was directly accessible, and did not require forwarding by a router. In this case, the Home Agent would not be able to intercept the packet.

Mobile IPv6 addresses this issue through a technique call *proxy neighbour discovery* (proxy ND). Neighbour Discovery [17] is a standard IPv6 protocol for the discovery of MAC addresses from IPv6 addresses, similar in concept to the ARP protocol for IPv4. Proxy ND involves an IPv6 node masquerading as another node at the MAC layer, by falsely responding to neighbour solicitations with its own MAC address. Home agents use proxy ND to ensure they intercept any IPv6 packets for a Mobile Node transmitted on its home network. To accomplish this, Home Agents also maintain a *proxy neighbour discovery table*, which contains the IPv6 addresses to which the Home Agent is acting as a proxy for. Entries to this table are added and removed as binding update messages with the 'Home Agent' flag set are added and removed from the binding cache.

2.6 Home Address Option

Mobile Nodes away from home have a choice of which global scope IPv6 address to use as a source for outgoing IPv6 packets. Either the node's home address could be used, or the current care-of address. However, neither of these choices are particularly desirable. If the current care-of address is used, then the source address for subsequent packets will change as a handoff takes place. This causes often irreparable problems for higher layer protocols such as TCP, which maintain transport layer identifiers and checksums based on network layer addresses. On the other hand, if the home address is used, then the outgoing IPv6 packet becomes susceptible to ingress filtering.

Ingress filtering is performed by many border routers to improve the security of the site to which they serve. Ingress filtering involves the inspection of the source address of all incoming IP packets, and verifying that the route to that address lies along the interface on which the packet was received. Any packets which fail this test are dropped as a security precaution. This can avoid many security attacks which use 'address spoofing'. Mobile Nodes sourcing their IPv6 packets with their home address on a foreign network can be mistakenly interpreted as a security threat by routers employing ingress filtering.

Mobile IPv6 defines a new IPv6 destination option, known as the *home address option*, which can provide a source address solution that is safe for transport protocols and is not susceptible to ingress filtering. This is achieved by a route optimised form of reverse tunnelling, which involves a level of minimal encapsulation when sending IPv6 packets from a Mobile Node. Table 4 illustrates the home address option.

IPv6 Header		Home Address Option	Transport Header	Payload
Source Address:	Care of Address	Home Address	TCP/UDP	Data
Dest. Address:	Correspon dent Node			
← 40 bytes →		← 18 bytes →		

Table 4 Mobile IPv6 Home Address Option

The Mobile IPv6 specification states that Mobile Nodes should source their IPv6 packets using a care-of address, thereby avoiding ingress filtering. However, any upper layer protocols should assume the source address of outgoing packets is the home address. All outgoing packets from a Mobile Node include a home address option. Upon receipt by a Correspondent Node, the address contained within the home address option replaces the source address of the packet, before any upper layer processing takes place.

2.7 Home Agent Discovery

Mobile IPv6 provides a mechanism for Mobile Nodes to automatically detect the presence of Home Agents on its home network. This mechanism involves all Home Agents joining the link local Home Agents anycast address. Mobile Nodes wishing to discover a Home Agent sends a binding update (with the 'H' flag set) to this anycast address. This message will be delivered to at most one Home Agent on the home network. Upon receipt of the message, the Home Agent responds with a binding acknowledgement, thereby informing the Mobile Node of the Home Agent's IPv6 address.

The binding updates sent to the Home Agents anycast address are otherwise ignored by Home Agents.

2.7.1 The Mobility Header

A new IPv6 extension header, the mobility header, is designed to contain the MIPv6 signalling messages. The mobility header is used by Mobile Nodes, Home Agents and Correspondent Nodes for all messaging related binding creation and management. The format of the mobility header is illustrated in Figure 4.

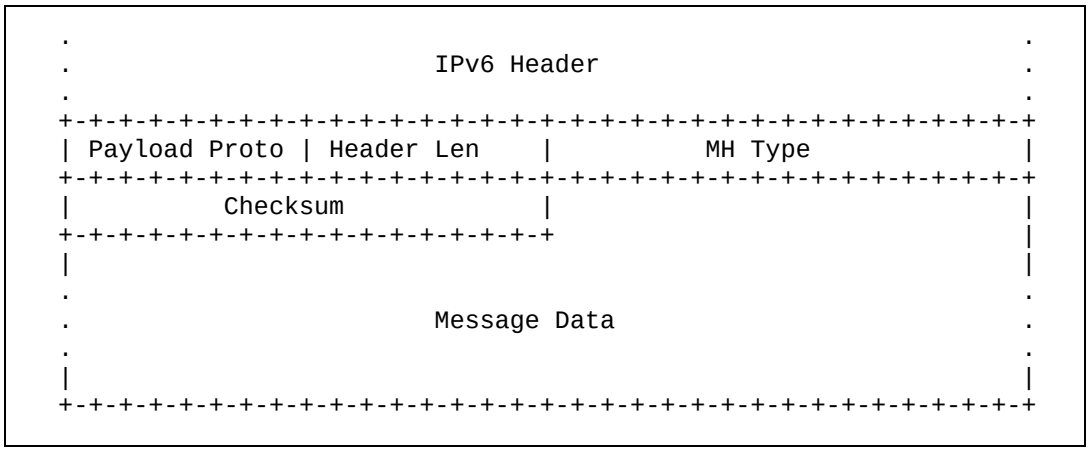


Figure 4 The Mobility Header Format

The mobility header is identified by a next header value of 62 in the IPv6 header (or an alternative preceding header if there is one). The 'MH Type' field identifies the specific mobility message in question and can be one of the following:

- Home Test Init (HoTI)
- Care-of Test Init (CoTI)
- Home Test (HoT)
- Care-of Test (CoT)
- Binding Request (BR)
- Binding Update (BU)
- Binding Acknowledgement (BA)
- Binding Missing (BM).

Due to the use of the new mobility header, the piggybacking of MIPv6 signalling with data is no longer possible. However, a separate extension to the protocol that will allow piggybacking in the presence of the mobility header may become available in the near future.

2.7.2 The Return Routability Method

The Return Routability (RR) method is a new binding update (BU) authorisation mechanism, suitable for use between Mobile Node and Correspondent Node peers, which have no previous

knowledge of each other. It is based on the principle of exchanging 'cookies' that verify the Mobile Node is 'alive' at its claimed address. The cookies are used by the Mobile Node to cryptographically protect the eventual BU message.

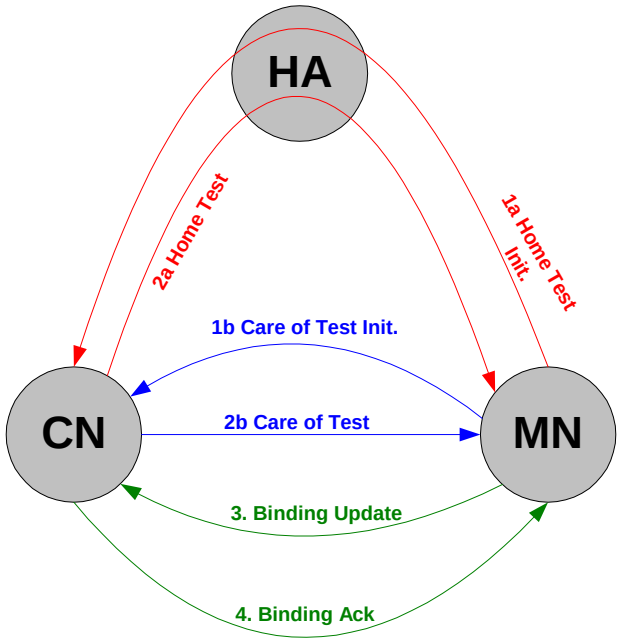


Figure 5 Return Routability Messaging

When a Mobile Node wishes to achieve route optimisation, it initiates the RR method as illustrated in Figure 5.

The HoTI and CoTI messages are sent simultaneously by the Mobile Node to the Correspondent Node.Upon the receipt of the HoTI and CoTI messages, the Correspondent Node computes two cookies based on the information contained in the messages, combined with its own secret key and nonce value. These cookies are inserted into the respective HoT and CoT messages, which are then sent back simultaneously to the Mobile Node.

Once the Mobile Node has received both the HoT and CoT messages, it has the cookies necessary to send the BU to the Correspondent Node. It hashes together the cookies to form a session key, which is then used to authenticate the BU that is sent to the Correspondent Node. When the Correspondent Node receives the BU, it can verify the information using its cookies and create a binding cache entry for the Mobile Node. The Correspondent Node may optionally acknowledge the BU with a BA.

3 Available Implementations

These are not all of the implementation available. However, these are the implementations we know of that are RFC 3775 compliant and are publicly available.

	MIPL	Cisco[1]	Microsoft[2]	KAME	HP-UX
Platform	Linux 2.6.8.1[a)]x	Cisco IOS	2000/XP/CE	FreeBSD	HP-UX-11i
Modes	MN/HA/CN	HA/CN	MN/HA/CN	MN/HA/CN	HA/CN
PND	Yes	Yes	Yes	Yes	Yes
IPv6-in-IPv6 tunnelling	Yes	Yes	Yes	Yes	Yes
DHAAD	Yes	Yes	Yes	Yes	Yes
Binding Management	Yes	Yes	Yes	Yes	Yes
HAO	Yes	Yes	Yes	Yes	Yes
Movement Detection	RAs	N/A	RAs and NDIS notifications	RAs	N/A
Smooth Handoff	Yes	Yes	Yes	Yes	N/A
IPsec	No (v1.1) Yes (v2.0)	No	Yes	Yes	Yes with HP-UX IPsec
Key exchange	MD5 or SHA-1	No	Manual	Manual	Unknown
Support for notebooks/ PDAs	Yes	N/A	Yes	Poor	N/A
MIPv6 built-in	No	Yes	No	Yes but not enabled by default	No. Is a component of TOUR 2.0
No. of patches	1	0	1 (XP and CE)	0	1
Set-up tools	mipdiag	command line tools	Auto-configuration and command line	Command line tools	Unknown
Licence	GNU	Commercial	Commercial	GNU	Commercial

Figure 6 Available Mobile IPv6 Implementations

Notes:

a) MIPL 2.0 RC1 has been tested in Redhat 9 and Fedora Core 2, with kernel version 2.6.8.1 with TAHI MIPv6 conformance test suite version 3.0b (HA), 3.0b3 (CN) and 3.0b4 (MN).

b) The MIPL Mobile IPv6 for Linux code version 2.0 RC1 is the 13th public release and is the first for the 2.6 kernel series. It supports IPsec protection of the home registration signalling, but is still missing tunnel mode protection of the HoTI/HoT signalling and tunneled payload data, as

[1] Available to CCO members.
[2] Not publicly available at time of writing but is expected to be soon.

well as MPS/MPA support. These features are announced to be included in the next release so that the implementation then should be fully RFC 3775 compliant.
Version 1.1 was the last release for the 2.4 kernel series.

4 Deployment Considerations

A general MIPv6 testbed would consist of:

- at least one Home Agent
- at least one Mobile Node
- at least one Correspondent Node
- several (>=2) IPv6 networks

At least one of the IPv6 networks must contain a Home Agent. Ideally, another two IPv6 networks should allow for the location of a Mobile Node (when away from home) and a Correspondent Node respectively. A very simple MIPv6 testbed like this is illustrated in Figure 7.

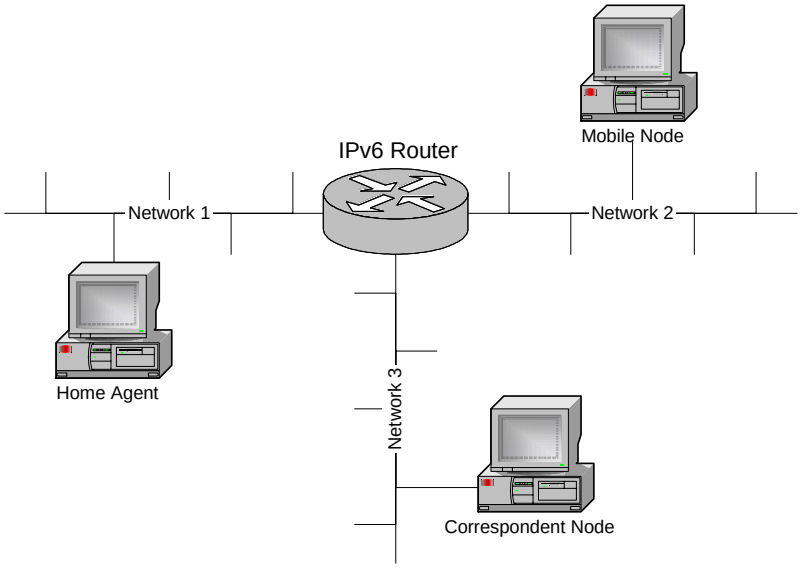


Figure 7 Simple Mobile IPv6 Testbed

The example in the figure shows three networks although, in theory, two networks would suffice (the Correspondent Node could be located in Network 1 or 2).

4.1 Hardware Requirements

In addition to the required network infrastructure (routers, cables, hubs, access points etc.) a MIPv6 testbed will require at least 3 separate machines: the Home Agent, the Mobile Node and the Correspondent Node. One could also use PC-based IPv6 routers (using open source systems like Linux and FreeBSD) instead of commercial IPv6 routers. This will certainly give more flexibility with regard to the addition of new IPv6 features and fine-tuning of network parameters (e.g. router

advertisement intervals). Another possibility is to use a PC-based router that also offers Home Agent functionality.

The simplest way to simulate handoffs between networks is to unplug the Ethernet (or other media type) cable to which the MN is currently attached and replace it with a cable from the network you wish to move into. Of course, this means that the Ethernet cables pertaining to the different networks should be within easy reach (e.g. within a network room) if quick handoff latencies are desired.

Alternatively, you may wish to employ Wireless LANs to better facilitate roaming between networks (see case studies in section 9). A simple configuration would be to have each network served by its own wireless access point (AP). In this way (assuming the APs are Wi-Fi certified), one can perform handoffs simply by moving between APs without the need to unplug the WLAN network adapter from the Mobile Node.

4.2 Software Requirements

Before configuring the MIPv6 testbed, you will need to make some policy decisions:

- What kind of global network prefixes you are going to use in your network? 6BONE prefixes (3ffe::/16) are now deprecated so you will need to obtain a suitable production prefix from the 2001::/16 address block by contacting your ISP.
- How are you going to implement the IP routing? In a small network `ripng` would be a good choice. In both, Linux and in FreeBSD, you can use `zebra` (http://www.zebra.org) for accomplishing this.

Since the MIPv6 specification is relatively new (only recently received RFC status) and is still somewhat work in progress, implementations tend to be staggered in terms of what features they support. Thus, interoperability between implementations is somewhat of a hit and miss affair. Therefore, it is wise to first deploy the same MIPv6 implementation on all the machines in the testbed (so make sure that the implementation has HA, MN and CN functionality). Once confidence in one implementation is achieved, you may decide to deploy other implementations and see if they are interoperable.

Note that movement detection algorithms rely on receiving router advertisements from the default IPv6 router when first entering a network. Intelligent movement detection algorithms will also make use of media disconnect/connect notifications and issue router solicitation messages to speed up the reception of these router advertisement messages. Therefore, it is essential that the routers pertaining to the networks in the MIPv6 testbed issue periodic router advertisements so that a MN can configure itself with a new Care of Address when it connects to the network.

The following sections will try to throw some light on how a MIPv6 testbed can be accomplished by using open source software like FreeBSD and Linux in addition to commercial software such as Microsoft Windows XP/CE and Cisco IOS.

5 Cisco Mobile IPv6

The Cisco MIPv6 implementation began with a research collaboration with Lancaster University and the porting of their MIPv6 implementation for Linux. Since this date, Cisco integrates the MIPv6 Home Agent support in a "Technology Preview" release that is available for experiment.

The feature is not offered yet in a commercial Cisco IOS release as Mobile IPv6 is still an immature protocol.

The Cisco IOS release of the protocol has been demonstrated at several events, e.g. at the Madrid IPv6 Summit, March 2002 and N+I Tokyo, July 2002. The configuration comprised a Cisco 2600 acting as Home Agent, the Mobile Node was a Compaq iPAQ running Linux, and the Correspondent Node was an AlphaServer running Tru64.

At the time of writing, the Cisco IOS release supports version 24 of the IETF draft specification. Based on Cisco IOS 12.2T release train, the Technology Preview works on Cisco 2600, 3600, 3700 and 7200 routers series.

5.1 Available feature set

Home Agent Home agent functionality will allow a suitably configured IPv6 router to act as a home agent for one or more mobile nodes when they are away from home.

Timer changes MIPv6 requires that the range of some timers be changed, for example, the minimum interval between unsolicited router advertisements is reduced from the 3 seconds specified in RFC2461. These relaxations are supported.

Advertisement Interval option MIPv6 requires that routers be configurable to allow them to send an Advertisement Interval option in their Router Advertisements to help mobile nodes perform movement detection. This feature is supported.

Duplicate Address Detection A mobile node may request that a home agent perform Duplicate Address Detection (DAD) when processing its registration and, whilst acting as the mobile node's home agent, defend its address. This feature is supported.

Dynamic Home Agent Address Discovery Home Agents in a subnet learn of each others presence and capabilities by listening to Router Advertisements. A mobile node may obtain the list of home agents on its home subnet by sending a request to the anycast address for all MIPv6-home-agents. This feature is supported.

Access Control List Support for an ACL to control the subnets from which the router will accept Binding Updates and DHAAD requests. This may be used to ‘black list’ certain sub-networks, preventing mobile nodes roaming to them, and refusing to perform correspondent node route optimization with mobile nodes that are currently visiting such sub-networks.

5.2 How to get it

At this date, the software is only available on the 'Ohanami' and can be obtained from Cisco CCO. Alternatively you may request access from Patrick Grossetete, pgrosset@cisco.com.

5.3 Installation

Appropriate Cisco IOS image must be downloaded on the router acting as Mobile IPv6 Home Agent. Minimum memory size must be compliant with the latest Cisco IOS 12.2T release, check CCO software centre to determine the appropriate memory size.

5.4 Configuration

5.4.1 Example Configuration

The following simple example configures a router as a Mobile IPv6 Home Agent on the interfaces Ethernet 1 and Ethernet 2. On the Ethernet2 interface, unsolicited router advertisements are configured to be transmitted every 0.05 seconds.

```
ipv6 unicast-routing
!
ipv6 mobile
!
interface Ethernet1
  ipv6 address 3000:1234:5678::1/64
  ipv6 mobile home-agent enable
interface Ethernet2
!
  ipv6 address 3000:1234:abcd::1/64
  ipv6 mobile home-agent enable
  ipv6 nd ra-interval msec 50
```

5.4.2 Configuration Commands

Global commands:

`[no] ipv6 mobile`

Enables Mobile IPv6. Default is disabled, all binding updates are ignored

`[no] ipv6 mobile mh-number <0-255>`

Changes the number used in the MIPv6 mobility header. The default is 62 which is also used by KAME.

`[no] ipv6 mobile bindings filter <acl>`

Configures a binding update filter using an ACL. When an ACL is configured all DHAAD requests and binding updates are filtered by Home Address and Destination Address. Default is no ACL configured.

This feature may be used to deny home agent services to mobile nodes that have roamed to particular sub-networks. When the filter blocks a binding update, a binding acknowledgement is returned with error status "Administratively prohibited". Default is no filter so all binding updates are accepted. Note that the filter is also applied to Dynamic Home Agent Address Discovery messages. When blocked, these are silently discarded.

In configuration of the ACL, the src is the CoA and the dst is the HoA.

`[no] ipv6 mobile binding lifetime <secs>`

Configures the maximum lifetime of a binding cache entry. Default is infinite lifetime.

`[no] ipv6 mobile binding max <int>`

Specifies the maximum number of registration bindings which may be maintained concurrently. By default, binding maximum is unset indicating unlimited. If the configured number of home agent registrations is reached or exceeded, subsequent registrations will be refused with the error "Insufficient resources". No existing bindings will discarded until their lifetime has expired, even if binding maximum is set to a value below the current number of such bindings.

`[no] ipv6 mobile binding refresh`

Default is 5 minutes (300 seconds).

Interface commands:

`[no] ipv6 mobile home-agent enable`

Enables home agent operation on the interface. By default, home agent operation is disabled.

`[no] ipv6 mobile home-agent preference <pref>`

Specifies the value to be use for Preference in the Home Agent Information Option transmitted on the interface. A value in the range -32768 to +32767 may be specified. By default, a value for Preference of zero is assumed for home agent operation on this interface.

`[no] ipv6 nd ra-interval <secs> | msec <msecs>`

Specifies the interval between sending unsolicited multicast Router Advertisements on this interface. The value for this should be set to less than or equal to the IPv6 Router Lifetime if this is a default router for the link. The optional sufffix `msec` has been introduced to indicate that the interval has been specified in milliseconds, rather than the default of seconds. This allows specification of the new minimum value of 0.05 seconds. The interval should be set to a low value on interfaces providing service to visiting mobile nodes in order to aid rapid movement detection. Note that to prevent undesirable synchronisation with other IPv6 nodes, the value may be randomly adjusted within +/- 20%.

`[no] ipv6 nd advertise-interval`

Specifies whether an Advertisement Interval option should be transmitted in Router Advertisements. This option may be used to indicate to a visiting mobile node how frequently it may expect to receive RAs. It may use this information in its movement detection algorithm.

```
[no] ipv6 nd prefix <prefix> | default
                    [ [<valid-lifetime> <preferred-lifetime>] |
                      [at <valid-date> <preferred-date>]
                      [off-link] [no-rtr-address] [no-autoconfig] ]
```

By default, all prefixes configured as addresses on the interface will be advertised in Router Advertisements. This command allows control over the individual parameters per prefix, including whether the prefix should be advertised or not. The `default` keyword can be used to set default parameters for all prefixes. A date can be set for prefix expiry. The valid and preferred lifetimes are counted down in real time. When the expiry date is reached the prefix will no longer be advertised. The keyword `no-rtr-address` means do not send the full router address in prefix advert; do not set the R bit.

Show commands:

`show ipv6 interface`

Existing command; output extended to include home agent data where and when applicable.

```
show ipv6 mobile binding [home-address <addr>]
                         [care-of-address <addr>]
                         [interface <int>]
```

Displays details of all bindings which match all the search criteria. If no parameters are specified, all bindings are listed. The output is in a form where parts may be cut/pasted into subsequent commands, e.g., `clear ipv6 mobile binding`.

`show ipv6 mobile globals`

Displays the values of all global configuration parameters associated with MIPv6, and lists the interfaces on which home agent functionality is currently operating.

`show ipv6 mobile traffic`

Displays binding updates received and binding acknowledgements sent.

`show ipv6 mobile home-agents [<interface> [<prefix>]]`

Displays the Home Agents List for the specified interface or, if none is specified, displays the Home Agents List for each interface on which the router is acting as a home agent. If a prefix is specified then only those addresses matching that value will be displayed.

Clear commands:

```
clear ipv6 mobile binding [home-address <prefix>
                          | care-of-address <prefix>
                          | interface <int>]
```

Clears all bindings with the mobile nodes which match the parameters. The parameter can be a prefix for the Care-of-Address or the Home-Address so that entire networks can be cleared. Use /128 to clear an individual node. The `interface` parameter will clear all bindings on the specified interface.

Note that when this command is used to delete bindings, the mobile node will not be informed that its home agent is no longer acting on its behalf.

```
clear ipv6 mobile home-agents [<interface>]
```

Clears the Home Agents List on the specified interface. The list will subsequently be reconstructed from received Router Advertisements.

```
clear ipv6 mobile traffic
```

Clears the statistics about the received binding updates and transmitted binding acknowledgements.

Debug commands:

```
[un]debug ipv6 nd
```

Existing command; output modified to include relevant home agent data.

```
[un]debug ipv6 mobile bindings-cache
```

Enables debugging of the bindings cache. Currently this only displays "SHUTDOWN" when the bindings cache is shutdown.

```
[un]debug ipv6 mobile forwarding
```

Enables debugging of soft tunnel forwarding.

```
[un]debug ipv6 mobile home-agent
```

Enables Home Agent debugging.

```
[un]debug ipv6 mobile registrations
```

Enables debugging of registrations, i.e. binding updates and binding acknowledgements.

```
[un]debug ipv6 mobile correspondent-node
```

Enables Correspondent Node debugging. This currently displays when a Home Address Options has been received.

5.5 Operation

Mobile IPv6 clients must be compliant with ID version 24 (or RFC 3775).

6 Mobile IPv6 for Linux

Mobile IPv6 for Linux (MIPL) is an implementation that was originally developed as a software project course in the Helsinki University of Technology (HUT), with the goal to create a prototype implementation of Mobile IPv6 for Linux. After the course, the implementation was further developed in the context of the GO/Core project at HUT Telecommunications and Multimedia Lab. It is an open source implementation, has been released under GNU GPL and is freely available to anyone.

The MIPL implementation has been tested in interoperability and conformance testing events such as the ETSI IPv6 Plugtests and TAHI Interoperability events.

6.1 How to get it

In Linux, the standard IPv6 stack is included in the mainstream kernel distribution. However, you have to patch the kernel if you want Mobile IPv6 features. Therefore, a MIPL release is usually compatible only with a specific kernel revision (unless between Linux revisions the kernel code that has to be patched has not changed). MIPL v1.1, which is described in this document, requires Linux kernel version 2.4.26. MIPL v1.1 is based on MIPv6 specification as described in RFC 3775 [1]. The most recent MIPL release, MIPL 2.0 RC1 requires Linux kernel version 2.6.8.1. It is recommended to use MIPL 1.1, at the time of writing MIPL 2.0 RC1 is a preview release and therefore very much still work in progress.

The kernel 2.4.26 can be obtained from your favourite kernel mirror site, e.g. http://www.kernel.org

The MIPL patch mipv6-1.1-v2.4.26tar.gz can be obtained from http://www.mobile-ipv6.org

6.2 Installation

The following steps describe the MIPL installation (based on the README and INSTALL files of the package).

1. Unpack the MIPL tar file, for example in the `/usr/src` directory. This will create the directory `/usr/src/mipv6-1.1-v2.4.26`
2. Go to the Linux source directory (e.g. `/usr/src/linux`)
3. Apply the MIPL kernel patch:
   `"patch -p1 < /usr/src/mipv6-1.1-v2.4.26/mipv6-1.1-v2.4.26.patch"`
4. "`make xconfig`" (or config or whatever you prefer) to configure the kernel: you should have the following settings in your configuration:

```
CONFIG_EXPERIMENTAL=y
CONFIG_SYSCTL=y
CONFIG_PROC_FS=y
CONFIG_MODULES=y
CONFIG_NET=y
CONFIG_NETFILTER=y
CONFIG_UNIX=y
```

```
CONFIG_INET=y
CONFIG_IPV6=m
CONFIG_IPV6_SUBTREES=y
CONFIG_IPV6_IPV6_TUNNEL=m
CONFIG_IPV6_MOBILITY=m
CONFIG_IPV6_MOBILITY_MN=m
CONFIG_IPV6_MOBILITY_HA=m
CONFIG_IPV6_MOBILITY_DEBUG=y
```

The last setting is optional. `CONFIG_IPV6_MOBILITY_DEBUG` turns on debugging messages. It is advised to set this to '`y`' in order to help debug any problems since MIPv6 is still very much an immature protocol.

5. Compile the kernel and modules:

   - `"make dep && make clean && make bzImage modules modules_install"`

6. Install the new kernel:

   Copy the created `bzImage` to `/boot/vmlinuz-2.4.24` and edit `/etc/lilo.conf` (or `/boot/grub/menu.lst` if you use GrUB) reflecting the changes, then run `lilo` to apply the changes (not applicable if you use GrUB).

7. Add the MIPv6 device:

   - `"mknod /dev/mipv6_dev c 0xf9 0"`

8. Build the user space tools of MIPL:

   - `"cd /usr/local/src/mipv6-1.1-v2.4.26"`
   - `"./configure"`
   - `"make && make install"`
   - the "`mipdiag`" tool should be available now, which allows configuration and status query.

9. If your distribution does not provide a router advertisement daemon, get one from http://v6web.litech.org/radvd/ and compile it.

10. Reboot into the new kernel.

11. Start MIPL with:
    `/etc/init.d/mobile-ip6 start`

6.3 Configuration

The configuration files for MIPL are as follows:

`/etc/network-mip6.conf`	- Mobile IPv6 features
`/etc/radvd.conf`	- Router advertisement daemon
`/etc/modules`	- Configuration file for modules loading

The main configuration file is `/etc/sysconfig/network-mip6.conf` although the path may be different in your particular Linux distribution (e.g. in Debian the path is `/etc/network/network-mip6.conf`). The following table lists the configuration variables that can be set in this file and their meaning.

Variable name	Description
FUNCTIONALITY	Defines the mode of the node: "ha" – Home Agent "cn" – Correspondent Node "mn" – Mobile Node Note that ha and mn both have cn functionality included
DEBUGLEVEL	If debugging was enabled when configuring the MIPL module, this variables controls the verbosity of the output (default: 0)
TUNNEL_SITELOCAL	Should unicasts to node's site-local address be tunnelled when mobile node is not in its home network (default: no)
MIN_TUNNEL_NR	Minimum number of free tunnel devices kept in cache on MN or HA. Must be set to at least 1 for MN and HA. To ensure successful bindings even during high work loads it could be set to a bigger value on the HA (default: 1).
MAX_TUNNEL_NR	Maximum number of free tunnel devices kept in cache on MN or HA. Must be set to at least 1 for MN and HA. To improve performance set it higher than MIN_TUNNEL_NR (default =3).
HOMEADDRESS	The MN's home address (including prefix length). Only valid in mn mode.
HOMEAGENT	The home agent address (including prefix length). Only valid in mn mode.

Figure 8 MIPL Configuration Parameters

6.3.1 Node Type

To configure a Home Agent, the following settings should be in `network-mip6.conf`:

```
FUNCTIONALITY=ha
MIN_TUNNEL_NR=1
MAX_TUNNEL_NR=5
TUNNEL_SITELOCAL=yes
```

For a Mobile Node, make sure you have the following entries in `network-mip6.conf`:

```
FUNCTIONALITY=mn
TUNNEL_SITELOCAL=yes
MIN_TUNNEL_NR=1
MAX_TUNNEL_NR=3
HOMEDEV=mip6mnha1
HOMEADDRESS=<the home address of the mn>
HOMEAGENT=<the address of the ha>
```

Note that if you only require CN functionality, you must configure the node as you would a MN.

During the early stages of deployment, it is recommended to increase the verbosity level of the kernel module in `network-mip6.conf` in order to to be able to debug any problems :

```
DEBUGLEVEL=4
```

For a HA you must have forwarding enabled in order for the HA to be able to forward captured packets to the MN when it is away from home:

```
"echo 1 > /proc/sys/net/ipv6/conf/eth0/forwarding"
```

This is assuming that you are using eth0 as the outgoing interface. In addition you should probably need to add a default route to the outgoing interface.

6.3.2 Route Optimisation and Return Routability

It is possible to control the use of route optimisation and return routability via the proc file system. If you do not wish to use route optimisation (and thus return routability) user the following command:

```
"echo 0 > /proc/sys/conf/net/ipv6/mobility/accept_return_routability"
```

The MN will then communicate with the CN only through the tunnel on the HA.. Route optimisation and return routability are enabled by default (setting is 1).

6.3.3 IPsec Authentication

Authentication and authorisation of Mobile IPv6 messages between the MN and HA with IPsec is not possible in 2.4 kernels as IPsec support is missing. However it is possible to use a 3rd party IPsec implementation with MIPL 1.1 to support IPsec authentication between the MA and HA.

6.3.4 Radvd

In order for the MN to be able to discover when it returns home, the HA must be configured to send out router advertisements. Do this by configuring `/etc/radvd.conf` with suitable parameters e.g.:

```
# cat /etc/radvd.conf
interface eth0
{
   AdvSendAdvert on;
   MaxRtrAdvInterval 3;
   MinRtrAdvInterval 1;
   AdvIntervalOpt off;
   AdvHomeAgentFlag on;
   HomeAgentLifetime 10000;
   HomeAgentPreference 20;
   AdvHomeAgentInfo on;
   prefix fec0:106:2700::2/64
   {
     AdvRouterAddr on;
     AdvOnLink on;
     AdvAutonomous on;
```

```
        AdvPreferredLifetime 10000;
        AdvValidLifetime 12000;
      };
    };
```

Start `radvd` with

`"/etc/init.d/radvd start"`

You can verify that RAs are being sent by running `radvdump`.

6.4 Usage Notes/Problems

Using earlier versions of MIPL we observed a few problems with medium to long term usage. In brief these concerned the kernel failing to add IPv6 routes and needing to be rebooted and unsolved negative binding acknowledgements sent by the HA on MN binding updates. However, with MIPL 1.0 and further versions we have not noticed these problems.

Multiple interface support works (we tested for example handovers between Bluetooth and 802.11 access networks with a MIPL client switching the priority of the interfaces), but sometimes when both interfaces receive router advertisements at short intervals (around 1 second), there are spurious switches between the interfaces, though the interface preference is not changed. This might be a problem with how the router advertisement lifetime is handled in the MIPL implementation. If the router advertisements are received at a slightly lower rate (2 seconds), these interface changes do not occur.

In our testbeds, we have not been able to test IPsec authentication between the MN and HA This is because authentication and authorization of MIPv6 messages between the MN and HA is missing in 2.4 series kernels. This will not be supported by the MIPL team in the 2.4 kernels unless someone backports the IPsec in 2.6 kernels to 2.4 kernels. However, MIPL for 2.6 series kernels (MIPL v2.0 onwards) will have IPsec support from the start.

Another problem which was observed is that the routing table is sometimes not correctly updated. So old routes from visited networks remain in the routing table.

7 KAME / FreeBSD Mobile IPv6

The KAME project developed a Mobile IPv6 implementation for the FreeBSD and NetBSD platforms. The code is implemented as part of the kernel. In addition to this, several user space programs have been developed for MIPv6 control, for extracting MIPv6 statistics and for dynamic home agent discovery

The implementation follows RFC 3775 [1] and includes functionality for HA, MAN and CN (mandatory for an IPv6 implementation that claims to be IPv6 compliant). In addition, KAME supports authentication of messages between a MN and its HA using IPsec [2].

7.1 How to get it

To get a proper Kame IPv6 stack proceed with the following steps (IPv6 is included in all standard BSD releases but for Mobile IPv6 the KAME Kernel is required):

- For FreeBSD, get and install *FreeBSD 4.10-RELEASE* or *5.3-RELEASE* (more information is available at http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/)
- For NetBSD, get and install NetBSD 2.0 (more information available at http://www.netbsd.org/guide/en/)
- Get the latest Kame SNAP from ftp://ftp.kame.net/pub/kame/snap/ or from mirror servers.

7.2 Installation

Since MIPv6 functionality is not enabled by default you need to enable the correct features and rebuild the kernel. Depending on the type of node you wish to install (MN, HA or CN) the entries in the kernel configuration differ.

The following features should be set in your kernel configuration file for MN functionality:

```
options         MIP6
options         MIP6_MOBILE_NODE
pseudo-device   hif  1
```

The following features should be set in your kernel configuration file for HA functionality:

```
options         MIP6
options         MIP6_HOME_AGENT
```

The following features should be set in your kernel configuration file for CN functionality:

```
options         MIP6
```

Compile and install the kernel. This will give you a kernel supporting the node type you configured. The next step is to compile the user space programs. This will also provide you with:

- `rtadvd` - a Mobile IPv6 suitable routing advertisement daemon

- `had` - a dynamic home agent discovery protocol daemon
- `mip6stat` - an application to gather MIPv6 related statistics
- `mip6control` - a control application for MIPv6 functions

The user space programs had, mipstat and mip6control are built automatically and should be installed in /`usr/local/v6/sbin` (check your path settings as well to make sure you load the kame versions of the applications instead of the standard ones `/usr/local/v6` has to be in your path before the path to the regular apps). The two supporting binaries "`mip6control`" and "`mip6stat`" should have been compiled by applying the ordinary kame tree "`make`" procedure.

To build rtadvd, you need to add the following line to its Makefile (this should be located at `/freebsd4/sbin/rtadvd/` in FreeBSD):

```
CFLAGS+=-DMIP6
```

Then recompile rtadvd and install it:

```
make clean
make
make install
```

7.3 Configuration

Configuration files you have to edit:

`/etc/rc.conf`	- system wide configuration file
`/etc/rtadvd.conf`	- configuration file for the router advertisement daemon.

For FreeBSD and NetBSD users, there is a special rc script that enables MIPv6 configuration to be specified in `/etc/rc.conf`.

If you are using FreeBSD, you must copy `rc` and `rc.mobileip6` in `/kame/freebsd4/etc` directory to the `/etc` directory.

If you are using NetBSD, you must copy the `/kame/netbsd/etc/rc.d/ mobileip6` file to the `/etc/rc.d` directory.

After this you can configure your node using `rc.conf`.

7.3.1 Home Agent

For having the Home Agent functionality on the FreeBSD system you will need to do the following (note that a node cannot act as a MN and HA at the same time). Add the following entries in `/etc/rc.conf`:

```
ipv6_mobile_enable="YES"
ipv6_mobile_config_dir="/usr/local/v6/etc/mobileip6"
ipv6_mobile_nodetype="home_agent"
ipv6_mobile_home_prefixes=<your home prefix>
ipv6_mobile_home_link=<your interface name>
```

Where `ipv6_mobile_home_prefixes` is your home prefix and `ipv6_mobile_home_link` is the interface name you use for the home network.

A home agent needs to contain routing functionality. Therefore you need other configuration parameters as would be required for a normal IPv6 router:

```
ipv6_gateway_enable="YES"
ipv6_router_enable="YES"
ipv6_router="/usr/local/v6/sbin/route6d"
ipv6_ifconfig_”xx”=<address of your interface>
```

The Access Router functionality is started at boot time if you add the following entries in the `/etc/rc.conf`:

```
rtadvd_enable=”YES”
rtadvd_daemon=/usr/local/v6/sbin/rtadvd
rtadvd_flags=”-ms -c /etc/rtadvd.conf”
```

Then you will have to adjust the file `/etc/rtadvd.conf` according to your needs.

```
#
# $Id: rtadvd.conf,v 1.2 2001/04/13 21:42:09 cco Exp $
# $Name:  $
#

#       $KAME$
#
# Note: All of the following parameters have default values defined
#       in specifications, and hence you usually do not have to set them
#       by hand unless you need special non-default values.
#
#       You even do not need to create the configuration file. rtadvd
#       would usually work well without a configuration file.
#       See also: rtadvd(8)

# that is advertised on each and every interface
default:\
             :chlim#64:rltime#10:rtime#10:retrans#0:\
             :vltime#30:pltime#10:mtu#1500:

# sent only by simple routers
router:\
        :raflags#0:pinfoflags#192:mininterval#3:maxinterval#5:\
        :tc=default:
```

```
# sent only by the Home Agents
ha:\
        :hatime#100:hapref#10:raflags#32:pinfoflags#224:\
        :mininterval#1:maxinterval#5:tc=default:

# per-interface definitions.
#   Mainly IPv6 prefixes are configured in this part.  However, rtadvd
#   automatically learns appropriate prefixes from the kernel's routing
#   table, and advertises the prefixes, so you don't have to configure
#   this part, either.
#   If you don't want the automatic advertisement, (uncomment and) configure
#   this part by hand, and then invoke rtadvd with the -s option.

# this is a router
xl0:\

:addrs#1:addr="3ffe:0400:0190:0046:250:4ff:fe64:eb78":prefixlen#64:tc=router:

# this is a HA interface
xl1:\

:addrs#1:addr="3ffe:0400:0190:0047:210:4bff:feb4:d3a1":prefixlen#64:tc=ha:
```

Figure 9: *Rtadvd sample configuration parameter file*

After rebooting, the FreeBSD box will act as a Home Agent and Access Router. For checking the status of your Home Agent you should use:

```
/usr/local/v6/sbin/mip6control
```

7.3.2 Mobile Node

To configure your node as a mobile node, add the following lines to `/etc/rc.conf`:

```
ipv6_mobile_enable="YES"
ipv6_mobile_config_dir="/usr/local/v6/etc/mobileip6"
ipv6_mobile_nodetype="mobile_node"
ipv6_mobile_home_prefixes=<your home prefix>
```

Replacing `ipv6_mobile_home_prefixes` with your home prefix.

7.3.3 Correspondent Node

To run a CN, no additional measures are required provided that the kernel was built with MIPv6 support (see above).

7.4 Remarks

In contrast to MIPL Kame supported IPsec from early on. MIPL version 1.1 is the minimum requirement for usage as a MN in conjunction with a Kame HA. Earlier versions of MIPL are incompatible with KAME HA. With MIPL 2.0 IPsec support should be included as well (not evaluated yet).

The KAME Mobile IPv6 stack had been developed for long time. Last year, for several reasons, it was decided to merge the implementation with that of the WIDE project[1], where another Mobile IPv6 code for BSDs had been developed by SFC in Keio University. Both partners agreed to unify and re-design their implementations. The resulting new MIP6 code is called *shisa*[2] which confirmed its interoperability during the latest ETSI IPv6 plugtest event.

The current shisa implementation is based on RFC3775, RFC3776, draft-ietf-nemo-basic-03 and draft-wakikawa-mobileip-multiplecoa-03. It works on FreeBSD 4.10, FreeBSD 5.3 and NetBSD 1.6.2. But note that FreeBSD 4.10 will not supported in kame snap soon as announced on snap-users mailing list.

[1] http://www.wide.ad.jp/
[2] http://www.mobileip.jp/

8 Microsoft Mobile IPv6 Technology Preview

The Microsoft Research (MSR) Mobile IPv6 implementation was produced in collaboration with Lancaster University as part of the LandMARC project [8]. During the project, Lancaster University's MIPv6 implementation for Linux was ported to the Windows 2000 operating system. The implementation was available in executable and source code format as a free download for research purposes from http://research.microsoft.com/mobileipv6/ . This MIPv6 implementation was a modified version of the MSR IPv6 stack, version 1.4 [7].

The public release of the MIPv6 implementation supported version 12 of the IETF MIPv6 draft and provided Mobile Node, Correspondent Node and Home Agent functionality. There was no public release of a MIPv6 implementation for other MS Windows operating systems. However, MIPv6 functionality in Windows XP and CE could be obtained from the Windows 2000 MIPv6 source code by re-compiling with appropriate changes made. Since then a technical preview for Windows XP and CE has been available to some Microsoft customers.

8.1 How to get it

Until recently, the MSR MIPv6 implementation was available for download from the MSR MIPv6 homepage [6]. Unfortunately, Microsoft is no longer making the implementation available for public download. The statement on the homepage reads:

"*Microsoft is keen to provide a supported implementation of Mobile IPv6 in its Windows products. We are working with our partners and customers, and with the IETF to provide this support. At the present time [December 2002], the IETF has not completed the Mobile IPv6 RFC. Microsoft does not therefore include Mobile IPv6 in its Windows products.*

A preview edition of Mobile IPv6 for early versions of Windows 2000, confirming to version 12 of the Mobile IPv6 Internet draft, has been available for research purposes from the Microsoft Research website. This has not been actively maintained against recent releases of the Windows operating system. We do not plan to upgrade this particular implementation because we do not plan to release further releases of IPv6 for Windows 2000. For the best experience with IPv6 from Microsoft, please use Windows XP and the forthcoming Windows .NET Server Family."

However, Windows XP and Windows .NET Server do not yet include MIPv6 functionality although they do have IPv6 functionality built in. Suffice to say, at the time of writing there is no publicly available MIPv6 functionality on a Windows operating system. However, there is a technical preview from Microsoft made available to certain customers. 6NET partners were able to gain access to this preview release. The following sections are intended as a guide for anyone that has a copy, or is able to obtain a copy of the technology preview.

The Mobile IPv6 Technical Preview is an unsupported implementation of the IETF Mobile IPv6 (draft 24) specification for Windows XP Pro SP1 and Windows CE .NET 4.2. It is provided in response to customer requests, permitting early experimentation and trials of the Mobile IPv6 (MIPv6) protocol.

The Technical Preview consists of a modified IPv6 protocol driver (tcpip6.sys) together with modified versions of the ipv6.exe, ipsec6.exe, and sagen.exe command line utilities. The ipv6.exe

and ipsec6.exe utilities can be used to configure a machine to act as a MIPv6 Correspondent Node (CN), Mobile Node (MN) or Home Agent (HA).

8.2 Installation

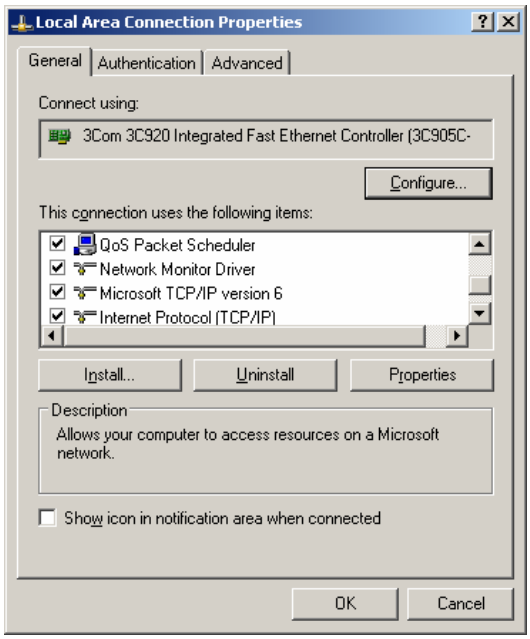
8.2.1 Pre-requisites

To install the technical preview on Windows XP, you must have Windows XP Service Pack 1 plus the Advanced Networking Pack.

For Windows CE .NET, you must be running CE version 4.2.

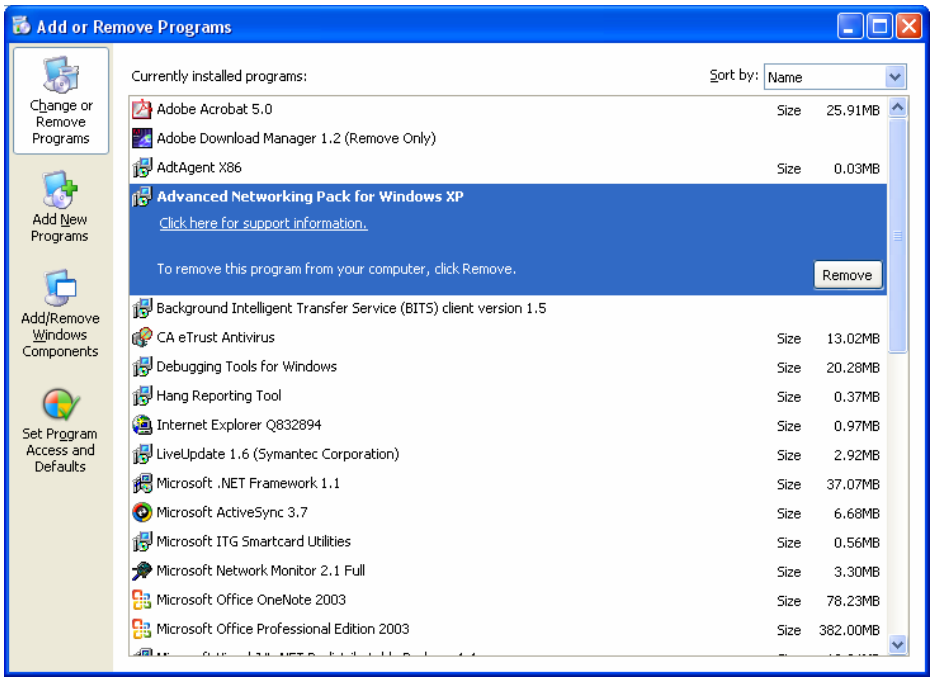
If you already have a working IPv6 stack on your Windows XP/CE machine, it is recommended to uninstall this before installing the MIPv6 technical preview. The MIPv6 technical preview includes base IPv6 functionality and will replace the old IPv6 stack on your machine.

To confirm if the stack is already installed select Start→Settings→NetworkConnections→ LocalAreaConnection→Properties. If "Microsoft TCP/IP Version 6" appears in the list of installed protocols, as shown below, then remove it by selecting it and choosing "Uninstall". You will have to reboot after uninstalling.



To verify that Service Pack 1 is installed, select Start→ControlPanel→SystemProperties→General and look for the text "Service Pack 1" under the "System:" part. Obtain and install Service Pack 1 if it is not already present.

To verify that the Advanced Networking Pack is installed, select Start→ControlPanel→Add or Remove Programs and looking for the text "Advanced Networking Pack", for example as shown below.



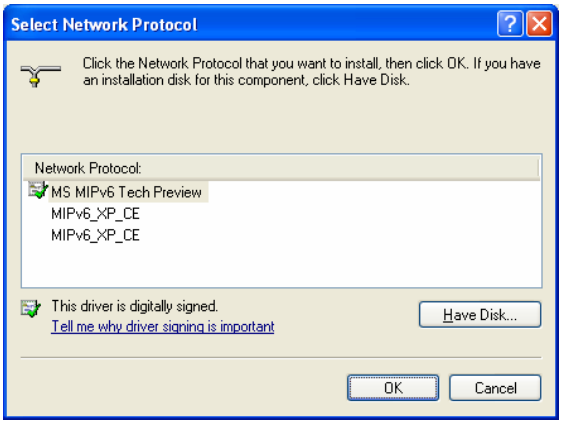
If the Advanced Networking Pack is not already installed, then visit the Windows Update site to obtain and install a copy. Note that the Advanced Networking Pack is listed as one of the optional (non-critical) updates available for Windows XP.

8.2.2 Installation Procedure for XP

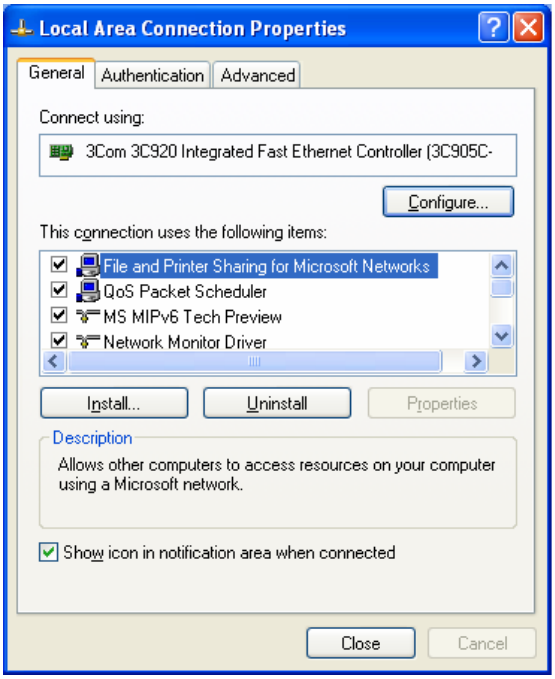
Start the install procedure by running `mipv6.exe` from the installation media.

The install procedure will initiate a reboot in order to replace iphlpapi.dll which is a widely used system file.

Once the Tech Preview has been installed you must manually activate the modified protocol driver. To do this select Start→Settings→NetworkConnections→LocalAreaConnection→Properties→ Install. Then choose "protocol" then "Add". Then select "MS MIPv6 Tech Preview", as shown below, from among the listed protocols displayed.



After clicking OK, you should be able to see the "MS MIPv6 Tech Preview" driver listed among the installed protocols.



As a simple post-installation test run `ipv6.exe gp`. If the installation has been successful you will see a number of parameters whose name is prefixed by the characters `MIPv6`. These parameters are unique to the MIPv6 versions of tcpip6.sys and ipv6.exe.

If you wish to uninstall the technical preview, you can do so via "Add or Remove Programs" in the Contro lPanel.

8.2.3 Installation Procedure for CE

Tranfer the appropriate .CAB file onto your CE device by using Microsoft® ActiveSync®[1] or a manual copy.
To begin the installation 'tap' the CAB file in file explorer. Select 'Device' as the target install location for '`Microsoft Corp. Mobile IPv6`', then tap 'Install'. When the 'Mobile IPv6 Setup' dialog appears press 'Ok'. This dialog is simply a reminder that a cold boot is required to uninstall this package.
After accepting the EULA, the package will copy the files into the \WINDOWS directory and then a setup completion dialog will appear.
After tapping 'Ok' on the setup completion dialog, you will need to 'warm boot' the device. How to do this will be depend on your PDA.
After the warm boot has completed, you can verify that Mobile IPv6 is installed by navigating to: 'Start->Settings->System->Remove Programs' and looking for '`Microsoft Corp. Mobile IPv6`' in the list of Programs in Storage Memory. Note that you should not actually use this method to uninstall

[1] http://www.microsoft.com/windowsmobile/downloads/default.mspx

MIPv6. Please do not attempt to remove the 'Microsoft Corp. Mobile IPv6' using this method because the TCP DLL is currently in use and will thus not uninstall.

Since the TCP DLL cannot be uninstalled from \WINDOWS while it is open, the normal CAB uninstall will not work. Therefore, a cold-boot (hard reset) is required. The down-side is that ALL other installed software is also lost and would need to be reinstalled after a cold boot
After the warm boot has completed you may need to run `sagen.exe` if you plan to use IPSec for authenticating messages between the MN and HA.

8.3 Configuration

The Mobile IPv6 stack can be dynamically configured to run in any combination of modes. These modes include Mobile mode, Correspondent mode, and Home Agent mode. When in Mobile mode, home addresses can be dynamically added and removed and the security settings for Home Agents can be configured. When any change takes place to the configuration, the new settings are stored in the Windows registry, where they are subsequently reloaded during driver initialisation.

In simple cases, the MIPv6 Auto-Configuration Service should succeed in automatically configuring the stack for Mobile Nodes, based on routers that advertise MIPv6 Home Agent service in their Router Advertisement messages. The operation of the MIPv6 Auto-Configuration Service can be controlled using the `MIPv6Conf.exe` utility. This utility can also be used to inspect and change the mobility parameters of the MIPv6 stack in a straightforward manner.

Beyond that, the MIPv6 code is configured using a new version of the `ipv6.exe` utility. The behaviour of the protocol driver is determined to some extent by global variables. The default values are sensible, but several of them can be changed using the `ipv6.exe` utility. The user can request that changes to specific global variables persist across reboots, in which case the parameter values are stored in the registry key:

`\\HKLM\SYSTEM\CurrentControlSet\Services\Tcpip6\Parameters`

By default, the MIPv6 stack assumes Mobile and Correspondent mode operation. Additional arguments (to those supplied in MSR IPv6 1.4) to the new version of `ipv6.exe` are:

ipv6.exe hau *h-addr ha-addr SPI SPD* Define home address *h-addr* using the Home Agent on address *ha-addr*. *SPI* is the IPSec Security Parameter Index of the inbound IPSec tunnel from HA to MN and `SPD` is the index of the outbound IPSec Security Policy that defines the (reverse) tunnel from the MN to its HA.

ipv6.exe hau *h-addr n ::0* Delete home address *h-addr*

ipv6.exe bc Inspect the state of the MIPv6 Binding Cache

ipv6.exe bu Inspect the state of the MIPv6 Binding Update List

ipv6.exe ha Inspect the state of the Home Address

ipv6.exe mip Inspect the mode (Mobile, Correspondent, Home Agent) of the MIPv6 stack

ipv6.exe mipu [MN] [CN] [HA] Set the mode of the MIPv6 stack to one or more of Mobile Node (MN), Correspondent Node (CN) or Home Agent (HA).

In particular ipv6 bc and ipv6 bu show which Care-of Address a mobile is using for each of its Home Addresses, and also shows the state of the Return Routability and Binding Update mechanisms. Sample output of these commands are shown below.

```
C:>\ipv6\etc>ipv6 bu

Home Address: 2000::2
  Host: 2000::3
    CoA      : 6/2001::240:96ff:fe27:cb02
    Expires  : 7s
    RRState  : ACTIVE
    TunnelBypassIndex: 12


  Host: 2000::1
    CoA      : 6/2001::240:96ff:fe27:cb02
    Expires  : 59s
    Flags    : HOME_AGENT
    RRState  : NO_RR ACTIVE


C:>\ipv6\etc>ipv6 bc

home: 2000::3
 c/o: 2001::240:96ff:fe36:6914
 seq: 11   Lifetime: 18s
 RRState : ACTIVE
```

The behaviour of the stack is controlled by number of parameters (listed below). The paraemters can be displayed and modified using the commands:

```
ipv6 gp
ipv6 gpu
```

`MIPv6Security`: off if the user has manually disabled security, default is on;

`MIPv6Mode`: acting as mobile node (MN), correspondent (CN) or Home Agent (HA);

`MIPv6RouteOptimize`: whether Route Optimization is enabled or disabled;

`MIPv6KcnInterval`: lifetime of each generation of the Kcn key, in seconds;

`MIPv6KcnGenerations`: number of generations of Kcn key;

`MIPv6HomeBindingLife`: initial lifetime of a home binding, in seconds;

`MIPv6RRBindingLife`: initial lifetime of an Return Routability binding, in seconds;

`MIPv6ErrorTimeout`: delay after receiving an error before next retry, in seconds;

`MIPv6HomeAgentPreference`: as advertised by DHAAD protocol;

`MIPv6SendMobilePrefixAdvertisements`: whether or not these are sent by HA;

`MIPv6InitialBindackTimeoutFirstReg`: see literal of similar name in spec.

The stack will remember mobility parameters, in particular home addresses, in the registry under key:

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip6\Mobility`.

These settings will survive reloads of the stack (via reboot or "`net [stop|start] tcpip6`") but are lost if the stack is reinstalled.

8.3.1 IPsec

The configuration of a Home Address on a mobile node will, in its normal recommended mode of secure operation, require that appropriate IPSec security associations already exist between a mobile node and its home agent. Specifically, we require an IPSec ESP SA to exist for protecting Mobility Headers whose end-points are the Mobile Node and the Home Agent. In addition, we require an ESP tunnel to exist between the Mobile Node and its Home Agent to protect the confidentiality of the Home Test Init and Home Test messages, else the Return Routability protocol can be compromised by an attacker who is monitoring the Mobile Node's foreign network.

Note that the Home Address configuration with IPsec does not persist across reboots. This is because the Microsoft production stacks do not persist IPsec state across reboots.

Since Windows does not have an implementation of IKE for IPv6, the IPSec configuration for the MIPv6 stack is installed manually through the `ipsec6.exe` program. Since manual configuration is somewhat laborious and error-prone, the `sagen.exe` command line utility is provided for generating the relevant IPSec configuration files (*.spd, *.sad) from a single table.

To use IPsec, a mobile node declares a home address using the command:

```
ipv6 hau HoA HA SPI SPD
```

where

- `HoA` is the new Home Address
- `HA` is the address of the Home Agent
- `SPI` is the IPSec Security Parameter Index of the inbound IPSec tunnel from HA to MN
- `SPD` is the index of the outbound IPSec Security Policy that defines the (reverse) tunnel from the MN to its HA.

This state may be inspected using the command:

```
ipv6 ha
```

If you do not wish to use IPsec, there is a way to disable the use of IPSec in the stack via the `MIPv6Security` global parameter. You can disable security with the command:

```
ipv6 gpu MobilitySecurity off
```

With this parameter set to off, no authentication is performed on home bindings and (reverse) tunnelling is done without IPSec thus leaving the Return Routability protocol vulnerable to anyone monitoring on a MN's foreign network.

8.3.2 Dynamic Home Agent Discovery

There is some limited support for the Dynamic Home Agent Discovery (DHAAD) and Home Prefix protocols. These protocols may be invoked manually from a Mobile Node using the commands:

```
ipv6 DHAAD probe <prefix>
ipv6 DHAAD <HomeAgent>
```

The results are displayed to the console.

8.3.3 MIPv6 Auto-Configuration Service

This WIN32 service program, running as a background service, listens for Home Agent advertisements on the network and auto-configures home addresses if desired.

This is achieved through event notification from the underlying Mobile IPv6 stack. As soon as the Mobile IPv6 stack receives a router advertisement with the Home Agent flag set, it notifies the auto-configuration service, which then checks whether a new home address is desired. If the number of active home addresses[1] is less than the maximum number of home addresses defined by the user, a new home address is configured (or simply proposed to the user when user-interaction is desired). Otherwise, the service simply updates the list of *Known Home Agents* held in the registry.

Each entry holds the number of router advertisements received from this Home Agent, the time stamp of the last router advertisement and the *rank* of the Home Agent. The rank is determined based on an adaptive control function, which takes the number of router advertisements, the last time stamp, and the previous rank as input parameters.

Note that auto-configured home addresses are built from the 64-bit prefix of the Home Agent network and the 64-bit EUI of the Mobile Node.

8.4 Operation

Once installed and configured, the MIPv6 functionality appears as an additional virtual interface via the `ipv6.exe` utility. For example:

```
C:\>ipv6 if
Interface 5: Ethernet: Local Area Connection
  Guid {BBAAAD13-A373-482C-BCED-0132170B3D72}
  uses Neighbor Discovery
  uses Router Discovery
  sends Router Advertisements
  forwards packets
  media reconnect flushes stale auto-configured state after 1500ms
  does not heuristically flush stale auto-configured state
  link-layer address: 00-01-02-b6-e8-e2
    preferred global 2001:630:80:7030::2, life infinite (manual)
```

[1] Home addresses for which the binding updates have been positively acknowledged by the Home Agent within a certain time frame.

```
    preferred link-local fe80::201:2ff:feb6:e8e2, life infinite
    multicast interface-local ff01::1, 1 refs, not reportable
    multicast link-local ff02::1, 1 refs, not reportable
    multicast link-local ff02::1:ffb6:e8e2, 1 refs, last reporter
    multicast link-local ff02::1:ff00:2, 1 refs, last reporter
    multicast interface-local ff01::2, 1 refs, not reportable
    multicast link-local ff02::2, 1 refs, last reporter
    multicast site-local ff05::2, 1 refs, last reporter
  link MTU 1500 (true link MTU 1500)
  current hop limit 128
  reachable time 39000ms (base 30000ms)
  retransmission interval 1000ms
  MaxRtrAdvInterval 600000ms
  MinRtrAdvInterval 200000ms
  DAD transmits 1
Interface 4: MIPv6 Pseudo-Interface
  Guid {BADE68B3-9FC9-5E9E-6285-D4F8E3E476DD}
  does not use Neighbor Discovery
  does not use Router Discovery
  media reconnect flushes stale auto-configured state after 1500ms
  does not heuristically flush stale auto-configured state
  link MTU 1280 (true link MTU 65515)
  current hop limit 128
  reachable time 23000ms (base 30000ms)
  retransmission interval 1000ms
  DAD transmits 0
Interface 3: 6to4 Tunneling Pseudo-Interface
  Guid {A995346E-9F3E-2EDB-47D1-9CC7BA01CD73}
  does not use Neighbor Discovery
  does not use Router Discovery
  media reconnect flushes stale auto-configured state after 1500ms
  does not heuristically flush stale auto-configured state
  routing preference 1
  link MTU 1280 (true link MTU 65515)
  current hop limit 128
  reachable time 26500ms (base 30000ms)
  retransmission interval 1000ms
  DAD transmits 0
Interface 2: Automatic Tunneling Pseudo-Interface
```

```
    Guid {48FCE3FC-EC30-E50E-F1A7-71172AEEE3AE}
    does not use Neighbor Discovery
    does not use Router Discovery
    media reconnect flushes stale auto-configured state after 1500ms
    does not heuristically flush stale auto-configured state
    routing preference 1
    EUI-64 embedded IPv4 address: 0.0.0.0
    router link-layer address: 0.0.0.0
    link MTU 1280 (true link MTU 65515)
    current hop limit 128
    reachable time 42000ms (base 30000ms)
    retransmission interval 1000ms
    DAD transmits 0
  Interface 1: Loopback Pseudo-Interface
    Guid {6BD113CC-5EC2-7638-B953-0B889DA72014}
    does not use Neighbor Discovery
    does not use Router Discovery
    media reconnect flushes stale auto-configured state after 1500ms
    does not heuristically flush stale auto-configured state
    link-layer address:
      preferred link-local ::1, life infinite
      preferred link-local fe80::1, life infinite
    link MTU 1500 (true link MTU 4294967295)
    current hop limit 128
    reachable time 15000ms (base 30000ms)
    retransmission interval 1000ms
    DAD transmits 0
```

The Technical Preview aims to implement the majority of the MIPv6 specification, with these exceptions:

- No support for dynamic keying of IPSec security associations.
- No support for dynamic notification of Home Address Prefix changes.
- Does not support stateful address configuration, which is not supported by the Microsoft IPv6 stacks.
- Does not support Multicast tunnelling, which is not supported by the Microsoft IPv6 stacks.
- No support for persistent Home Addresses -- the Microsoft IPv6 stacks do not persist IPSec state across reboots and IPSec configuration is a prerequisite to Home Address initialisation.
- Unable to send high frequency (30ms - 70ms) Ras because of the grain of the internal 500ms timer used by the production IPv6 stacks.

The technical preview passes the majority of the TAHI MIPv6 compliance test suite version 2.2 and 2.0.13 (but not all).

Experience at Lancaster demonstrated that the implementation works well in accordance with draft version 24 and exhibits relatively stable behaviour. However, experience with handoff testing showed that the implementation has relatively poor handoff latencies in the range of 2 to 30 seconds depending largely on the behaviour of network adapters and frequency of router advertisements. There can also be quite erratic behaviour when two MNs move simultaneously.

9 Case Studies

9.1 Fraunhofer Fokus

Since the last version of this deliverable the Fokus Mobile IPv6 testbed has undergone several changes. The current environment does not longer distinguish between an internal and an external part as before when there was a local, experimental testbed not constantly connected to the 6net network and another part with continous provision of native IPv6 connectivity to the outer 6NET world. Furthermore the test environment was completed by several components providing for VoIP- and video conferencing capabilities.

The Mobile IPv6 testbed consists of the following components:

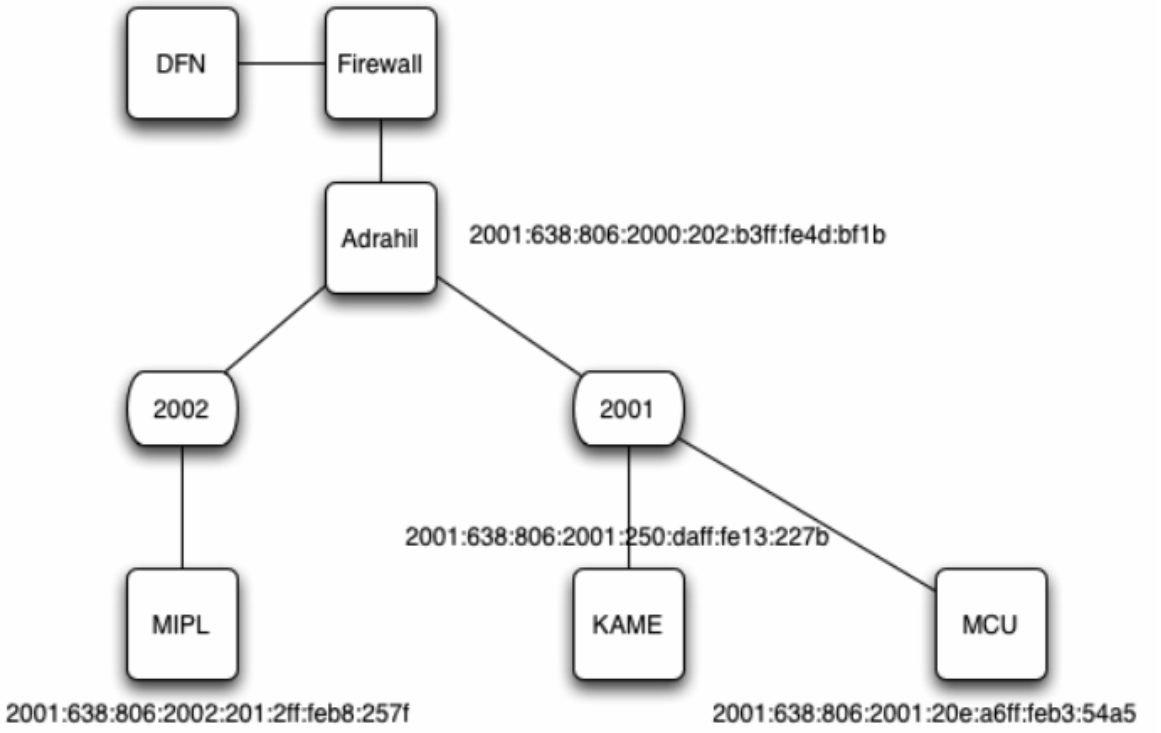


Figure 10 Schematic representation of the testbed setup

Connected to the central router “adrahil” there are basically two different networks with prefixes 2001:638:806:2002::/64 and 2001:638:806:2001::/64. Attached to those networks are the corresponding Home Agents for the MIPL- and the KAME-Mobile IP implementation, respectively. A MCU connected to the “2001”-network was used as Corresponding Node for Mobile IPv6 functionality- and interoperability testing: Using a MN with video conferencing equipment, i.e. the GnomeMeeting application with camera and headset, this scenario allowed for

establishing a connection to the CN-MCU and subsequently changing the IPv6 network point of attachment while maintaining the connection to the MCU.

Currently the Fokus testbed is made up of the different components:

- End systems with different operating systems
  At the leaves of the network, standard PCs with different operating systems (Windows Server 2003, Windows XP, Linux, FreeBSD) are installed. They are used for testing IPv6 network applications like video conferencing, web surfing, downloading audio and video streams, IP telephony applications etc.
- Home Agents
  Mobile IPv6 Home Agents as offered by the MIPL- and KAME project.
- IP softphone
  The original BonePhone application was complemented for performance reasons by the SIP-based KPhone telephony application (also developed in the context of WP5 of the 6NET project) for use as MN.
- FreeBit hardphone for use as MN.
- GnomeMeeting video conference client for use as MN.
- H.323 OpenMCU for use as CN.

MIPL-HA

The Linux HA is running version 1.1 of the MIPL implementation. The HA has the address: 2001:638:806:2002:201:2ff:feb8:257f. The home address in the network has the prefix: 2001:638:806:2002::/64.

Kame-HA

The KAME HA is running a snapshot dated 22/03/2004. Due to compatibility reasons you need at least version 1.1 of the MIPL implementation if you want to use the KAME HA in combination with a Linux mobile node. The HA has the address: 2001:638:806:2001:250:daff:fe13:227b.The home address in the network has the prefix: 2001:638:806:2001::/64.

MCU-CN

The MCU is running version 1.1 of the MIPL implementation as well, the address is 2001:638:806:2001:20e:a6ff:feb3:54a5. There are two rooms, the default room is “room101”. In order to enable checking of local audio and video settings there is a loopback room simply called “loopback”. In case of audio or video problems the MCU room “loopback” can be used to check proper working of audio and video peripherals.

IPsec

IPSec was turned off at the time of MIPv6 testing due to compatibility reasons. Only the KAME implementation supported it at that time.

9.1.1 Testbed components

Host	System	OS	Type	IPv6 Address
Trolloc	Cisco 7206VXR[1]	Version 12.2(8)T4	Router	DFN - 6WIN - IPv6 - Network prefix usage for FhG Fokus: 2001:0638:0806::/48
adrahil	PIII/500 MHz	Debian GNU/Linux Rel. 2.4.18, dual stack	Access router IPv6	2001:638:806:2000:202:b3ff:fe4d:bf1b 2001:638:806:2001:202:b3ff:fe4d:c8e0 2001:638:806:2002:250:4ff:fe64:eb78
mipl.lab6.fokus.fraunhofer.de			HA (MIPL)	2001:638:806:2002:201:2ff:feb8:257f
kame.lab6.fokus.fraunhofer.de			HA (KAME)	2001:638:806:2001:250:daff:fe13:227b
mcu.lab6.fokus.fraunhofer.de (MCU)		GNU/Linux 2.4.26 i686	CN	2001:638:806:2001:20e:a6ff:feb3:54a5

Table 5: IPv6 testbed components

9.1.2 Offering a Virtual Home Network Service

One can think of a scenario in which a Mobile Node is roaming only between foreign networks, never arriving home.

For offering this kind of service one needs a HA connected to the public IPv6 infrastructure. The MN's home address has to have the same prefix as the HA's one.

The Fokus testbed offers this kind of use of MIPv6 Home Agents for functionality and interoperability tests.

The required configuration parameters for the concerning MN are:

In case you want to use the Linux HA: MIPL HA-address: mipl.lab6.fokus.fraunhofer.de (2001:638:806:2002:201:2ff:feb8:257f) Home address in the network: 2001:638:806:2002::/64

In case you want to use the Kame HA: KAME HA-address: kame.lab6.fokus.fraunhofer.de (2001:638:806:2001:250:daff:fe13:227b) Home address in the network: 2001:638:806:2001::/64 (Note: To use the Kame HA with a Linux Mobile Node (MN) you need at least version 1.1 of the MIPL-implementation.)

[1] Cisco 7206VXR, 6-slot chassis, 1 AC Supply w/IP Software, 7200VXR NPE-400 (128MB default memory), 256 MB Memory for NPE-400 in 7200 Series, 1-Port Packet/SONET OC3c/STM1 Singlemode (IR) Port Adapter, Cisco 7200 Input/Output Controller with Dual 10/100 Ethernet, Cisco 7200 I/O PCMCIA Flash Disk, 128 MB Option, Cisco 72009 Series IOS IP

In order to support simple MN-CN connectivity/roaming tests an IPv6 H.323-MCU was connected to the “2001”-network: The address of the MCU is: mcu.lab6.fokus.fraunhofer.de (2001:638:806:2001:20e:a6ff:feb3:54a5) There are two conferencing rooms, the default room is "room101". For the convenience of checking local audio settings an audioloopback room called "loopback" was set up. If you encounter any kind of audio problems in "room101", please use the room "loopback" first to check if your local microphone and speaker settings work correctly.

There is no security association or any other restriction whatsoever. Any Mobile Node in the network above should be able to use these HAs.

9.2 Lancaster University

Lancaster University has had a Mobile IPv6 testbed in one form or another since 1997. In those early years we used Lancaster University's own implementation running on Linux. Since then, more extensive research efforts (some with the help of Lancaster University to port their code to different platforms) have spawned other implementations which are much more up to date and robust than the Lancaster University implementation. In our MIPv6 testbed for 6NET we have thus not considered the Lancaster University implementation since it is now not interoperable with more recent implementations.

9.2.1 The Testbed

As simplified diagram of the Lancaster University MIPv6 testbed is shown in Figure 11. Connectivity to 6NET (via SuperJANET) is achieved through a Cisco 7206 router. The testbed is then divided into Virtual Local Area Networks (VLANs) with each subnet (generally a ::/60 or a ::/64) comprising its own VLAN. The Extreme Summit Ethernet switch is capable of assigning VLAN tags according to IPv6 prefixes and switching them accordingly (thus, it acts as a virtual IPv6 router). All of the IPv6 VLANs are terminated on the Cisco 7206.

The 802.11 wireless LAN is configured so that each access router (i.e. HA) of the wireless LAN is analogous to one ESSID. Only one Access Point per ESSID is illustrated in the figure, although wireless coverage for each ESSID can be extended with more Access Points. Of course, configuring multiple Mobile IPv6 networks in this manner will eat up the available 802.11b channel space rather quickly. The figure above illustrates how multiple Mobile IPv6 HAs can have their associated Access Points arranged so that a maximum of 3 overlapping channels is seen in one cell footprint. Obviously, this is rather straightforward to accomplish in a lab environment where our main focus is to investigate the interactions between multiple Mobile IPv6 networks. In a real (semi)production environment the close geographical proximity of different distinct Mobile IPv6 networks is much less likely

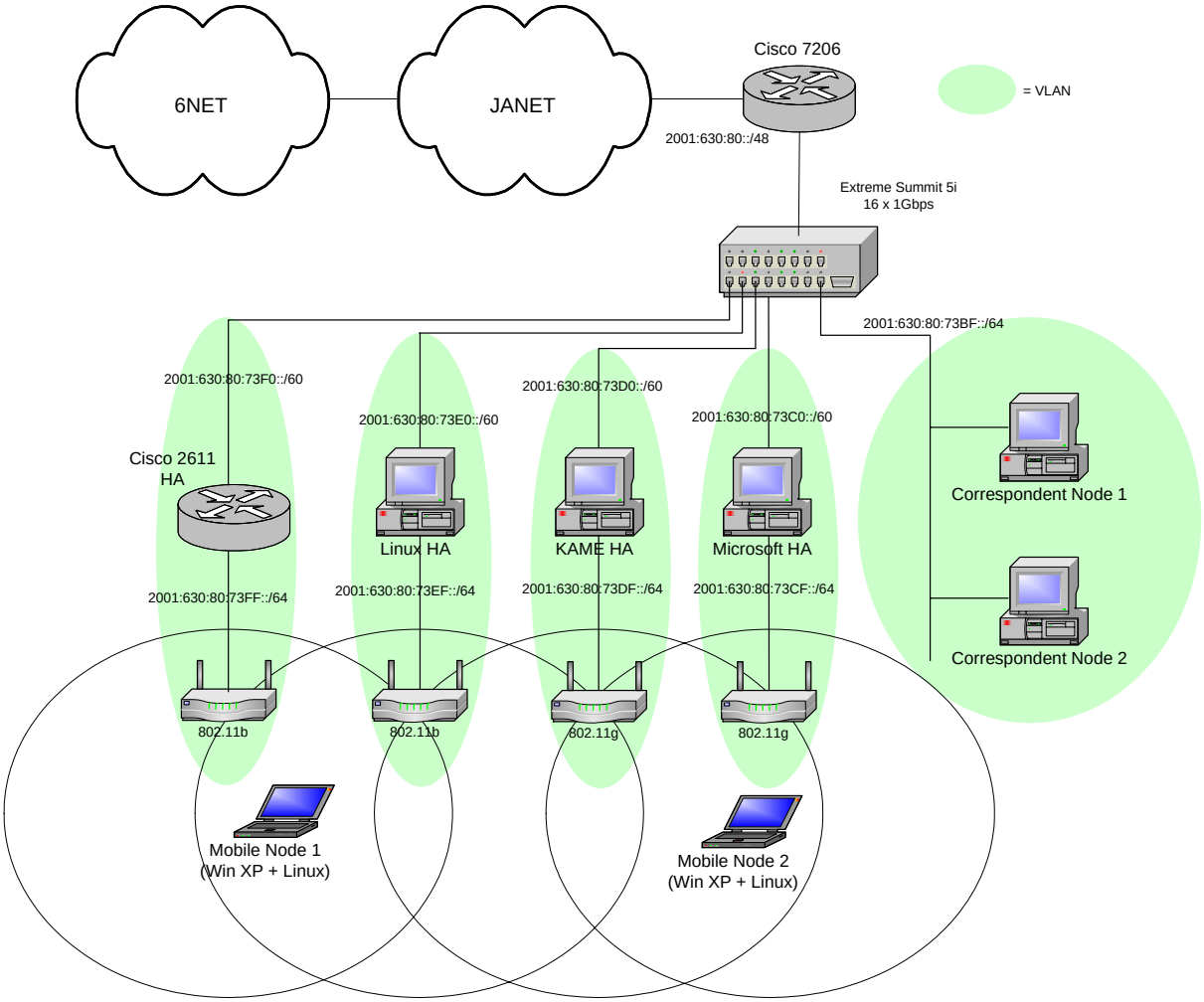


Figure 11 ULANC MIPv6 Testbed

9.2.2 Components

The various components in our Mobile IPv6 testbed are as follows.

Home Agents

The testbed employs four different Home Agents. One using the Microsoft MIPv6 technical preview implementation for Windows XP, another using MIPL v1.1, another using the Cisco Ohanami EFT and finally one Home Agent using KAME. The Microsoft, Linux and KAME Home Agents are all PC based routers each comprising AMD Athlon XP 2.6Ghz CPU 512MB RAM. The Cisco Home Agent is a 2611 XM with 128MB on board RAM.

Mobile Nodes

The Mobile Nodes in the testbed are dual-boot (Windows XP and Linux 2.4.26) machines and can therefore use either MIPv6 implementation of the two available operating systems. The Windows

XP operating system has Service Pack 1 installed along with Microsoft's Advanced Networking Pack update. The MIPv6 implementation on the Windows XP operating system

Correspondent Nodes

The two PC-based Correspondent Nodes comprise AMD Athlon XP 1.6Ghz CPUs and 256MB RAM. One of the PCs is a dual-boot Windows XP and Redhat Linux system, the other is a FreeBSD system running FreeBSD 4.10-RELEASE. Note that the Mobile Nodes may also perform the role of a Correspondent Node.

The make-up of the various components in our Mobile IPv6 testbed are summarised in Table 6.

	Hardware	System	MIPv6 Implementation
Cisco HA	2611 XM, 128MB RAM	Ohanami IOS EFT	Included in IOS
Microsoft HA	AMD Athlon XP 2.6Ghz, 512MB RAM	Windows XP SP1 + advanced networking update	Draft v24 compliant Technical Preview
KAME HA	AMD Athlon XP 2.6Ghz, 512MB RAM	FreeBSD 4.10-RELEASE	SNAP /kame/snap/kame-20050103-freebsd410-snap.tgz
Linux HA	AMD Athlon XP 2.6Ghz, 512MB RAM	Redhat Linux 2.4.26	MIPL v1.1
Mobile Node 1	Sony Vaio Intel Pentium III Mobile 1Ghz, 256MB	Dual boot Windows XP SP1 with advanced networking update and Redhat Linux 2.4.26	Draft v24 compliant Technical Preview MIPL v1.1
Mobile Node 2	Dell Latitude C610 Pentium III Mobile 1Ghz, 256MB	Dual boot Windows XP SP1 with advanced networking update and Redhat Linux 2.4.26	Draft v24 compliant Technical Preview MIPL v1.1
Correspondent Node 1	AMD Athlon XP 1.6Ghz, 256MB	Free-BSD 4.10 – RELEASE	SNAP /kame/snap/kame-20050103-freebsd410-snap.tgz
Correspondent Node 2	AMD Athlon XP 1.6Ghz, 256MB	Dual boot Windows XP SP1 with advanced networking update and Redhat Linux 2.4.26	MIPL v1.1

Table 6 Lancaster MIPv6 Testbed Components

9.2.3 Addressing and Subnetting

The rationale behind the addressing and subnet allocations is as follows. The prefix allocated to Lancaster University by JANET is 2001:0630:0080::/48.

Other than some special addresses reserved for e.g. router interface numbering and DNS, Lancaster University IPv6 addresses observe the following format:

<48 UNI> <1 Res> <3 Site> <12 Subnetting> <64 Host>

where:

<48 UNI> - 48 bit University prefix 2001:630:80::/48

<1 Res> - 1 reserved bit for future aggregation

<3 Site> - 3 bit code identifying the site of the network

<12 Subnetting> - 12 bits for site subnetting

<64 Host> - 64 bit host identifier

All of the sites, with the exception of site 7 (Research and Development), use their 12 bits for subnetting as follows:

<8 Building> <4 Network>

where:

<8 Building> - 8 bit code identifying a building within the site

<4 Network> - 4 bits to allocate subnets within each building

However, for our experimental Mobile IPv6 testbed we assigned it to site 7, research and development. We felt it was wise to have production and experimental research networks attributed to different sites and our Mobile IPv6 testbed is very much non-production traffic. The prefix for research and development networks is:

2001:630:80:7000::/52

Site 7, research and development uses its 12 bits for subnetting as follows:

<2 Reserved> <6 Networks> <4 Subnets>

where:

<2 Reserved> - 2 bits reserved for future aggregation

<6 Networks> - 6 bits to allocate to R&D networks

<4 Subnets> - 4 bits to allocate Subnets

Hence the full address format for addresses on the research and development network is:

<48 UNI> <1 Res> <3 Site> <2 Reserved> <6 Networks> <4 Sub-Networks> <64 Host>

There are 6 bits to allocate research and development networks (2^6 = 64 networks) and they are allocated in a flat manner.

For example:

2001:630:80:700::/60 - Network 0

2001:630:80:701::/60 - Network 1

2001:630:80:702::/60 - Network 2

2001:630:80:703::/60 - Network 3

…

2001:630:80:73F::/60 - Network 63

Further subnets can be defined arbitrarily based on the 4 bits to allocate for subnets.

So each Mobile IPv6 network is assigned a ::/60 prefix as illustrated in Figure 11. Currently each Mobile IPv6 network comprises one Home Agent. Each HA will advertise a ::/64 prefix via Router Advertisements to hosts on its link. Currently there is no stateful address configuration (e.g. using DHCPv6) on our MIPv6 networks and all hosts use stateless address autoconfiguration (remember that not every host on the HA's link needs to be a MN). A MN will know that it is on its home network when it sees the RA from its HA. It is possible to have multiple HAs per network but we chose this topology as it facilitates easier debugging during testing.

9.2.4 Testing

We have tested manually configured IPsec security associations between Microsoft MN and KAME HA. This seems to work well in that the Microsoft MN is able to register with the KAME HA and binding updates are performed successfully. When testing a Linux MN or HA we have only used non- IPsec authenticated communication between the MN and HA. The same is true when using a Cisco HA. In general we have found that all the implementations in our testbed are interoperable in a basic form.

For obvious reasons we do not employ any AAA or access control mechanisms when performing handover latency tests. This is especially important when using any streaming applications as the delay incurred when waiting for network access to be granted can result in a considerable number of lost packets. We have found that using IPerf [19] between the CN and MN is a very useful tool for discovering the extent of packet loss during handovers when comparing different network and different tunings of the RA intervals.

Since each Home Agent also acts as the default router on its respective link as shown in Figure 11 we configure each Home Agent to announce unsolicited Router Advertisements at more frequent intervals than the default. In general, we use a Router Advertisement interval of 1 second which is sufficient for most adaptable TCP applications such as email, web, ftp etc. However, we have noticed considerable (i.e. unpleasant) disruption when using a 1 second interval when streaming video and/or audio. Reducing the interval to around 300ms seems to be fair benchmark to set for supporting streaming media. Although it is possible to reduce the interval even further, in most tests we did not observe enough improvement in the stream disruption to justify the extra link bandwidth and Home Agent CPU cycles being used.

Initially, the most simple way of testing the operation of the MIPv6 testbed is by testing basic connectivity between the Mobile Node and Correspondent Node with a stream of ping requests from the Correspondent Node to the Mobile Mode while the Mobile Node moves between its home network and a foreign network. In a correctly configured Mobile IPv6 testbed, the ping requests should continue to be answered as the Mobile Node moves. Sometimes we observe one ping timing out as the packet is lost. This can occur if the ping request is sent just as the Mobile Node becomes

unreachable in its original location but has not yet completed the binding update procedure to be reachable at its new location. However, on a properly configured testbed we rarely observe more than one ping timeout.

For testing that TCP connections survive movement of the Mobile Node, a simple test is to run telent or similar (using the client and server in either combination of Mobile Node and Correspondent Node) and observe that the session remains active after movement. You should not observe any noticeable effects (save some possible interruption in character echo on the terminal) if all is configured correctly.

Beyond these basic tests some users may want to experiment with the TAHI compliance test suite described at [18].

9.3 University of Oulu

University of Oulu has carried out experiments with Mobile IPv6 for Linux in heterogeneous wireless networks. Especially the handover performance has been studied.

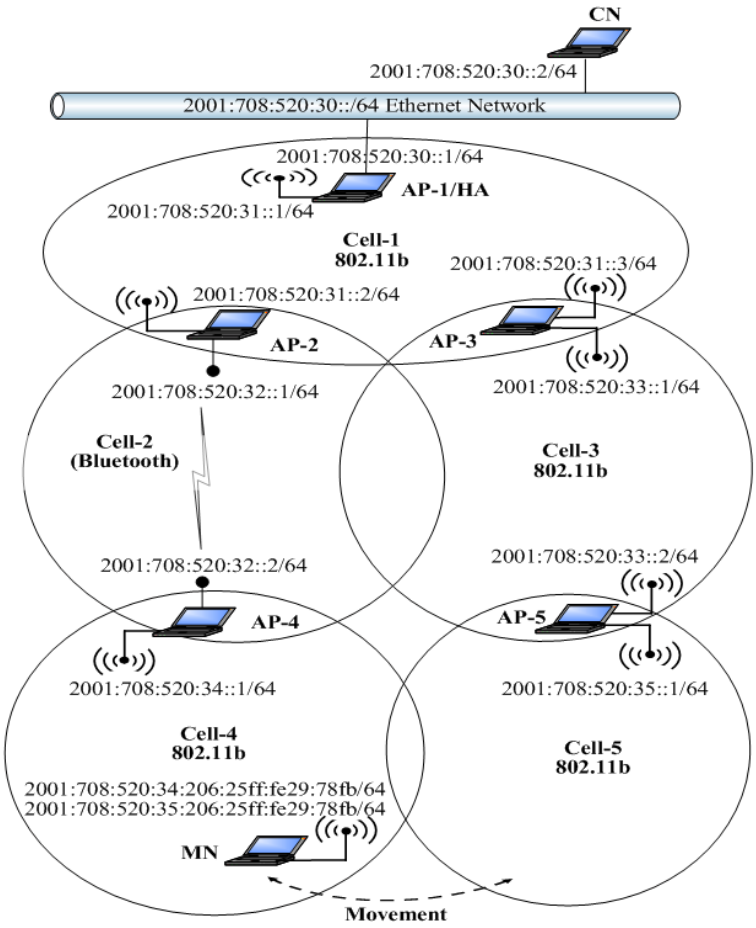


Figure 12 University of Oulu heterogeneous wireless MIPv6 testbed

9.3.1 Handover Performance

This series of experiments investigates the effect of handover duration on the TCP disruption time. We will describe handoffs between AP-4 and AP-5 (i.e. between the fast path and the slow path) as shown in Figure 12. AP-4 and AP-5 generate one router advertisements every 1.5 seconds. We define the handover time as the amount of time starting from when the mobile node can not get any packets from its old access point until the time when it receive binding acknowledgement from its home agent via the new access point. The first handoff experiment is from AP-5 to AP-4 (i.e. from the fast path to the slow path). The MN starts sending TCP traffic to the CN using AP-5 and moves towards AP-4. Figure 13shows the time sequence number plot of the TCP connection including the handover time. We found that it takes around 4.9 seconds to handoff from AP-5 to AP-4. The TCP disruption time was found to be 10.5 seconds. As illustrated in the figure, the TCP goes through 4

consecutive timeouts and retransmissions. Because of TCP exponential backoff mechanism, TCP doubles the size of its timeout interval each consecutive timeouts. Following the handover, the TCP waits for the last timeout to elapse before starts sending data using the new care-of-address. This result demonstrate that the handover duration contribute approximately 40% of the TCP disruption time. The remaining time delay is due to TCP congestion control mechanism. The average handover duration and TCP disruption time are shown in Table 7.

Handover from-to	Average handover duration [seconds]	Average TCP disruption time [seconds]
From fast to slow path	3.94	8.40
From slow to fast path	1.49	3.91

Table 7 Handover duration and TCP disruption time.

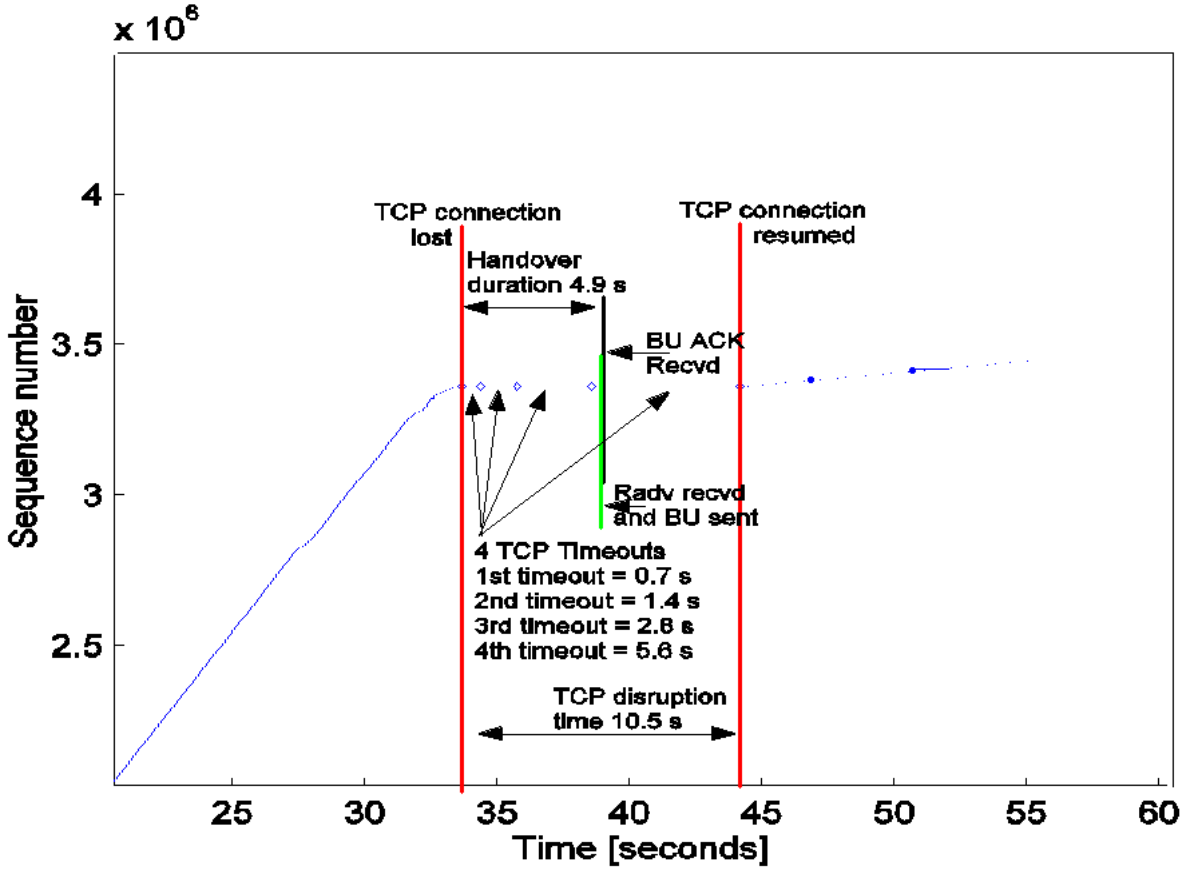


Figure 13 TCP packet trace during handover from AP-5 to AP-4

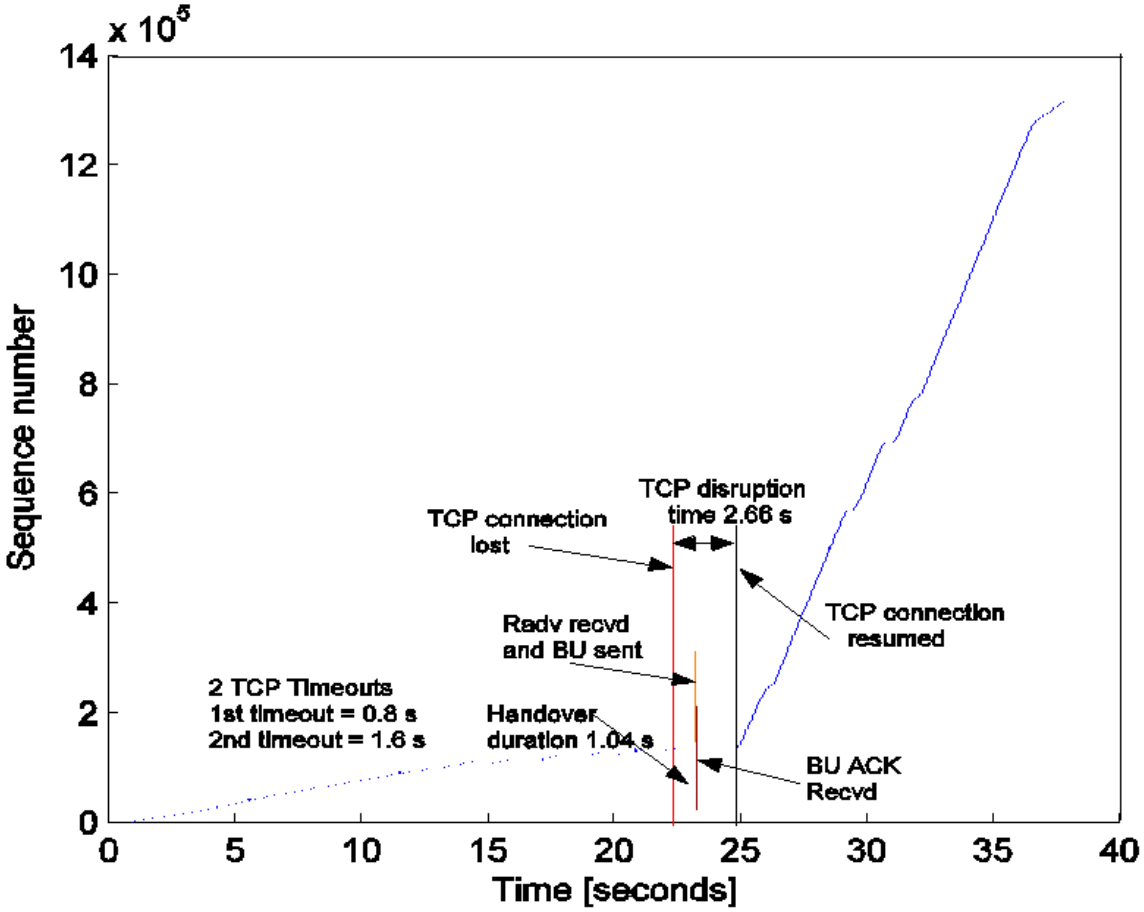


Figure 14 TCP packet trace during handover from AP-4 to AP-5.

The same behaviour was noticed when the handover was performed from AP-4 to AP-5 (i.e. from the slow path to the fast path). However, handover duration significantly decreased to about 1.04 seconds. The TCP disruption time takes around 2.66 seconds. Figure 14shows the TCP behaviour during a handover from AP-4 to AP-5. It can be seen that fewer timeouts and retransmissions have occurred in this experiment. The overall average handover duration and TCP disruption time are shown in Table 7. This results indicate that handover between AP4 to AP5 is faster than from AP5 to AP4. This is because related signalling packets travel faster in the fast path. It also shows that TCP performs well when handover is from a low data rate to a high data rate network. However, due to the TCP slow start mechanism, the recovery is slow when the handover is from the high data rate to the low data rate path. After the handover, the MN starts sending packets at slow rate (sets the congestion window to 1 MSS) and increases its sending rate exponentially fast. Several methods have been introduced to optimize the Mobile IPv6 handover process, including hierarchical MIPv6 and fast handover. The fast Handover protocol has been proposed as away to minimize the interruption in service experienced by a Mobile IPv6 node as it changes its point of attachment to the Internet. Using the fast handover mechanism would allow the MN to send and receive packets from the time that it disconnects from one access point to the time it registers a new care-of address from the new access point.

10 References

[1] D. Johnson, C. Perkins, J. Arkko, "Mobility Support in IPv6," IETF RFC 3775, June 2004

[2] J. Arkko, V. Devarapalli, F. Dupont, "Using IPsec to Protect Mobile IPv6 Signaling between Mobile Nodes and Home Agents," Internet Draft: draft-ietf-mobileip-mipv6-ha-ipsec-06, IETF, June 30, 2003

[3] KAME homepage, http://www.kame.net

[4] SHISA homepage, http://www.mobileip.jp/

[5] Mobile IPv6 for Linux homepage, http://www.mipl.mediapoli.com/

[6] Microsoft Research Mobile IPv6 homepgae, http://research.microsoft.com/mobileipv6/

[7] Microsoft Research IPv6 homepage, http://research.microsoft.com/msripv6/

[8] The LandMARC homepage http://www.landmarc.net/

[9] Lancaster University MIPv6 package, http://www.cs-ipv6.lancs.ac.uk/ipv6/MobileIP/

[10] Peter Bieringer's Linux IPv6 HOWTO, http://www.bieringer.de/linux/IPv6/

[11] 6WIND homepage, http://www.6wind.com

[12] Debian GNU/Linux homepage, http://www.debian.org/

[13] The Cisco IPv6 Statement of Direction, available for download via http://www.cisco.com/warp/public/732/Tech/ipv6/ipv6_learnabout.shtml

[14] G. O'Shea, M. Roe, "Child-proof Authentication for MIPv6 (CAM)", Computer Communication Review, Vol. 31, No. 2 (April 2001)

[15] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", IETF RFC 2460, December 1998

[16] C. Perkins, "IP Mobility Support for IPv4 (revised)", IETF RFC 3220, January 2002

[17] T. Narten, E. Nordmark, W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", IETF RFC2461, December 1998

[18] The TAHI Project Homepage, http://www.tahi.org/

[19] IPerf Homepage, http://dast.nlanr.net/Projects/Iperf/

Abbreviations

AP	Access Point
BA	Binding Acknowledgement
BU	Binding Update
CAM	Child-proof Authentication for MIPv6
CN	Correspondent Node
CoT	Care of Test
CoTi	Care of Test Init
DHAAD	Dynamic Home Agent Address Discovery
HA	Home Agent
HAO	Home Address Option
HoA	Home Address
HoT	Home Test
HoTI	Home Test Init
MN	Mobile Node
PND	Proxy Neighbour Discovery
RA	Router Advertisement
RS	Router Solicitation