Implementation and Evaluation of Dual Stack Mobile IPv6

Koshiro Mitsuya[1], Ryuji Wakikawa[1], and Jun Murai[1]

Keio University, Japan. {mitsuya,ryuji,jun}@sfc.wide.ad.jp

Abstract. Dual Stack Mobile IPv6 (DSMIPv6) is an extension of Mobile IPv6 to support IPv4 care-of address and to carry IPv4 traffic via bi-directional tunnels between mobile nodes and their home agents. Using DSMIPv6, mobile nodes only need the Mobile IPv6 protocol to manage mobility while moving within both the IPv4 and IPv6 Internet. This is an important feature for IPv6 mobility during its deployment phase because IPv6 access network is not widely deployed yet. This paper describes the DSMIPv6 implementation on BSD operating systems and presents results of the experiments using the implementation.

1 Introduction

Mobility support in IPv6 is important, as mobile computers are likely to account for a majority or at least a substantial fraction of the population of the Internet during the lifetime of IPv6. Hence, the IETF has standardized Mobile IPv6 [1] in 2004 and NEMO Basic Support [2] in 2005, to address host and network mobility.

Mobile IPv6 and NEMO allow mobile nodes (host and router) to move within the Internet while maintaining IP reachability and ongoing sessions, using a permanent IPv6 address for the host or a permanent IPv6 prefix inside the moving network. In these schemes, a mobile node keeps a home address and a mobile network prefix, which are permanent, and an IP address acquired from the network it is visiting, which is called care-of address. A special redirection server called home agent maintains the mappings between the home address and the care-of address, and between the home address and the mobile network prefix. The home agent intercepts packets on behalf of the mobile node and sends them to the mobile node's care-of address when the mobile node is away from its home network; thus the ongoing sessions can be kept alive.

Mobile IPv6 and NEMO are now in the deployment phase and there are two issues. First, it is acknowledged that mobile nodes will use IPv6 addresses only for their connections and will, for a long time, need IPv4 home addresses that can be used by upper layers, since IPv6 applications are not widely deployed. Second, as IPv6 wireless access networks are not widely deployed, it is also reasonable to assume that mobile nodes will move to a network that does not support IPv6 and therefore need the capability to support an IPv4 care-of address.

The Dual Stack Mobile IPv6 (DSMIPv6) specification [3] extends the Mobile IPv6 capabilities to allow mobile nodes to request their home agent, to forward IPv4/IPv6 packets addressed to their home addresses, to their IPv4/IPv6 care-of address(es). Using DSMIPv6, mobile nodes only need Mobile IPv6 or NEMO Basic Support to manage mobility while moving within the Internet; hence eliminating the need to run both IPv6 and IPv4 mobility management protocols simultaneously.

This paper describes the DSMIPv6 implementation on SHISA [4, 5], a Mobile IPv6 and NEMO implementation on BSD operating systems, and its evaluation. Considerations for the current specification are also discussed in this paper.

This paper is organized as follow. We give an overview of DSMIPv6 in Sec. 2 followed by an overview of SHISA in Sec. 3. We then present the design of our implementation in Sec. 4 and the implementation details in Sec. 5. We performed experiments by using our implementation and the results and considerations are reported in Sec. 6. This paper concludes in Sec. 7.

2 Dual Stack Mobile IPv6

This section presents an overview of the Dual Stack Mobile IPv6 (DSMIPv6).

2.1 Overview

A node supporting both IPv4 and IPv6 is called a dual stack node. It is important to develop dual stack nodes under IPv6 deployment phase because we cannot rapidly make the transition to IPv6. In fact, many applications are still using IPv4 and most of the access networks support only IPv4. The situation is the same for Mobile IPv6 (MIPv6) deployment. Mobile node will visit IPv4 only access networks and will use IPv4 only applications in the deployment phase of MIPv6.

In order to use MIPv6 by dual stack nodes, mobile nodes need to manage an IPv4 and IPv6 home or care-of address simultaneously and update their home agents' bindings accordingly. This concept is shown in Fig. 1. On the figure, MN is a mobile node, HA is a home agent, and CN is a correspondent node. MN1 is connected to an IPv6 network and MN2 is connected to an IPv4 network.

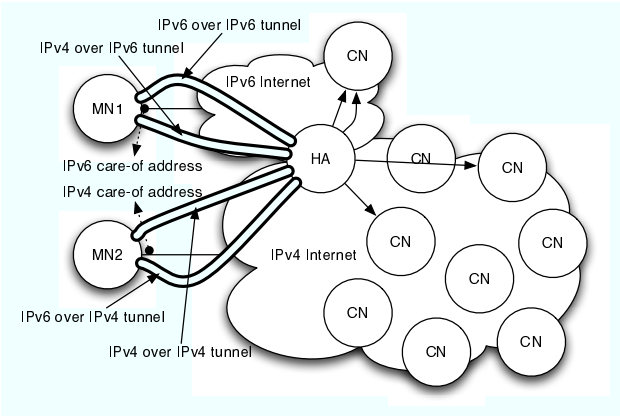

Fig. 1. The concept of DSMIPv6

A MN has both IPv4 and IPv6 home addresses. HA is a dual stack node connected to both IPv4 and IPv6 Internet. As MN1 is visiting IPv6 network, MN1 configures a global unique IPv6 address as its care-of address. MN1 registers the care-of address to HA, and both IPv4 and IPv6 home addresses bound to the address. IPv4 traffic goes through IPv4 over IPv6 tunnel between MN1 and HA, and IPv6 traffic goes through IPv6 over IPv6 tunnel. In a similar way, MN2 registers IPv4 care-of address. Traffic goes through IPv6 over IPv4 tunnel or IPv4 over IPv4 tunnel. By this way, mobile nodes need only MIPv6 to manage mobility while moving within both IPv4 and IPv6 Internet.

We give details of DSMIPv6 operation in the rest of this section. However, Home Agent Address Discovery feature and Network Address Translator (NAT) Traversal feature are not mentioned because they are not implemented yet.

2.2 Binding Management

A mobile node needs to update the bindings on its home agent with its current care-of address. If the mobile node is a dual stack node and has an IPv4 and IPv6 home address, it creates a binding cache entry for both addresses. The format of the IP packet carrying the binding update and acknowledgment messages varies depending on the visited network. IPv6 network is always preferred than IPv4 network, so there are three different scenarios to consider with respect to the visited network:

1. The mobile node configures a global unique IPv6 as its care-of address.
2. The mobile node only configures a global unique IPv4 address as its care-of address.
3. The mobile node only configure a private IPv4 address as its care-of address.

The operation for case 1 is explained in Sec. 2.3 and the operation for case 2 is explained in Sec. 2.4. Case 3 is not explained in this paper as mentioned in Sec. 2.1.

2.3 Visiting IPv6 Global Foreign Network

In this case, the mobile node configures a global unique IPv6 address as its care-of address (V6CoA). The mobile node sends a binding update message (BU) to the IPv6 address of its home agent (V6HA), as defined in [1]. The binding update message includes the IPv4 home address option, which is defined in [3]. The packet format is shown in Fig. 2. The packet format of the normal MIPv6 binding update message is also shown in the figure for comparison.

```
MIPv6 BU:
  IPv6 header (src=V6CoA, dst=V6HA)
    Destination option
      HoA (IPv6 home address)
    Mobility header
      BU

DSMIPv6 BU:
  IPv6 header (src=V6CoA, dst=V6HA)
    Destination option
      HoA (IPv6 home address)
    Mobility header
      BU [IPv4 home addres]
```

Fig. 2. Binding update message formats

```
MIPv6 BA:
  IPv6 header (src=V6HA, dst=V6CoA)
    Routing header type 2
      HoA (IPv6 home address)
    Mobility header
      BA

DSMIPv6 BA:
  IPv6 header (src=V6HA, dst=V6CoA)
    Routing header type 2
      HoA (IPv6 home address)
    Mobility header
      BA [IPv4 addr. ack.]
```

Fig. 3. Binding Acknowledgement message formats

After receiving the binding update message, the home agent creates two binding cache entries, one for the mobile node's IPv4 home address and one for the mobile node's IPv6 home address. Both entries will point to the mobile node's IPv6 care-of address. Hence, whenever a packet is addressed to the mobile node's IPv4 or IPv6 home addresses, it will be forwarded via the bi-directional tunnel to the mobile node's IPv6 care-of address. Effectively, the mobile node establishes two different tunnels, one for its IPv4 traffic (IPv4 over IPv6) and one for its IPv6 traffic (IPv6 over IPv6) with a single binding update message.

After binding cache entries are created, the home agent sends a binding acknowledgment message (BA) to the mobile node as defined in [1]. If the binding update message included an IPv4 home address option, the binding acknowledgment message includes the IPv4 address acknowledgment option. The packet format is shown in Fig. 3. This option informs the mobile node whether the binding was accepted for the IPv4 home address. The packet format of the normal MIPv6 binding acknowledgment message is also shown in the figure for comparison.

2.4 Visiting IPv4 Only Global Foreign Network

In this scenario, the mobile node needs to tunnel IPv6 packets containing a binding update message to the home agent's IPv4 address (V4HA). The mobile node uses the IPv4 care-of address (V4CoA) as a source address in the outer header.

The binding update message contains the mobile node's IPv6 home address in the home address option. However, since the care-of address in this scenario is the mobile node's IPv4 address, the mobile node must include its IPv4 care-of address in the IPv6 packet. The IPv4 address is represented in the IPv4-mapped IPv6 address format (V4MAPPED) and is included in the source address field of the IPv6 header. If the mobile node has an IPv4 home address, it also includes the IPv4 home address option. The packet format is as shown in Fig. 4.

```
DSMIPv6 BU (IPv4):
  IPv4 header (src=V4CoA, dst=V4HA)
    UDP header
      IPv6 header (src=V4MAPPED, dst=V6HA)
        Destination option
          HoA (IPv6 home address)
        Mobility header
          BU [IPv4 home address]
```

Fig. 4. Binding update message format in IPv4 network

After accepting the binding update message, the home agent will update the related binding cache entry or create a new binding cache entry if such entry does not exist. If an IPv4 home address option was included, the home agent will update the binding cache entry for the IPv4 address or create a new entry for the IPv4 address. Both binding cache entries point to the mobile node's IPv4 care-of address.

All packets addressed to the mobile node's home address(es) (IPv4 or IPv6) will be encapsulated in an IPv4 header that includes the home agent's IPv4 address in the source address field and the mobile node's IPv4 care-of address in the destination address field.

After creating the corresponding binding cache entries, the home agent sends a binding acknowledgment message. If the binding update message included an IPv4 home address option, the binding acknowledgment message includes the IPv4 address acknowledgment option as shown in Fig. 5. The binding update message is encapsulated in an IPv4 payload whose destination is the IPv4 care-of address (represented as an IPv4-mapped IPv6 address in the binding update message).

```
DSMIPv6 BA (IPv4):
  IPv4 header (src=V4HA, dst=V4CoA)
    [UDP header] (if NAT is detected)
      IPv6 header (src=V6HA, dst=V4MAPPED)
        Routing header type 2
          HoA
        Mobility header
          BA ([IPv4 addr. ack.], NAT DET)
```

Fig. 5. Binding acknowledgement message format in IPv4 network

3 SHISA: Mobile IPv6 and NEMO Implementation on BSD operating systems

As our DSMIPv6 implementation is an extension of SHISA, we give an overview of the SHISA stack in this section.

SHISA [4, 5] is an open source Mobile IPv6 and NEMO Basic Support implementation on BSD operating systems. It has been developed by the KAME project [6].

The design feature of SHISA is to separate packet forwarding functions and signal processing functions into the kernel and user land programs. The operation of Mobile IPv6 and NEMO Basic Support is mainly IP packet routing (forwarding or tunneling). In order to obtain better performance, the packet routing has been implemented in the kernel space. The signal processing has been implemented in the user land space because it is easier to modify or update user land programs then the kernel. This is important because the signaling processing of Mobile IPv6 and NEMO Basic Support is complicated. This separation provides both good performance and efficiency when developing the stack.

In SHISA, a mobile node (host or router) consists of small kernel extensions for packet routing and several user land daemons (MND, MRD, BABYMDD, NEMONETD, HAD and CND). MND/MRD are daemons which manage bindings on a mobile node. BABYMDD is a daemon which detects the changes of care-of addresses and notifies MND/MRD of the changes. NEMONETD is a daemon which manages bi-directional tunnels. HAD is a daemon which manages bindings on a home agent. CND is a daemon which manages bindings on a correspondent node. Depended on the node type, one or several SHISA daemons run on a node.

Fig. 6 shows the relation of the SHISA modules. The objects with solid line are modules implemented for SHISA. The dotted line objects are modules existing in the original BSD system and the shaded ones are modified for SHISA.

SHISA programs communicate with the kernel and other programs using the Mobility Socket [7]. The Mobility Socket is a special socket to exchange the mobility related information between the kernel and user land programs.

The Neighbor Discovery module and the Address Management module notify the movement detection daemon (BABYMDD) of the changes of IP address through the Routing Socket.

The signaling messages such as binding updates and acknowledgments are exchanged between the SHISA programs on two nodes, for example, MRD on a mobile router and HAD on its home agent. The binding information retrieved from the signal exchange is stored in the user land space as the Binding Update database for mobile nodes and the Binding Cache database for home agents or correspondent nodes. Only subsets of the databases that are necessary for packet forwarding are installed into the kernel.

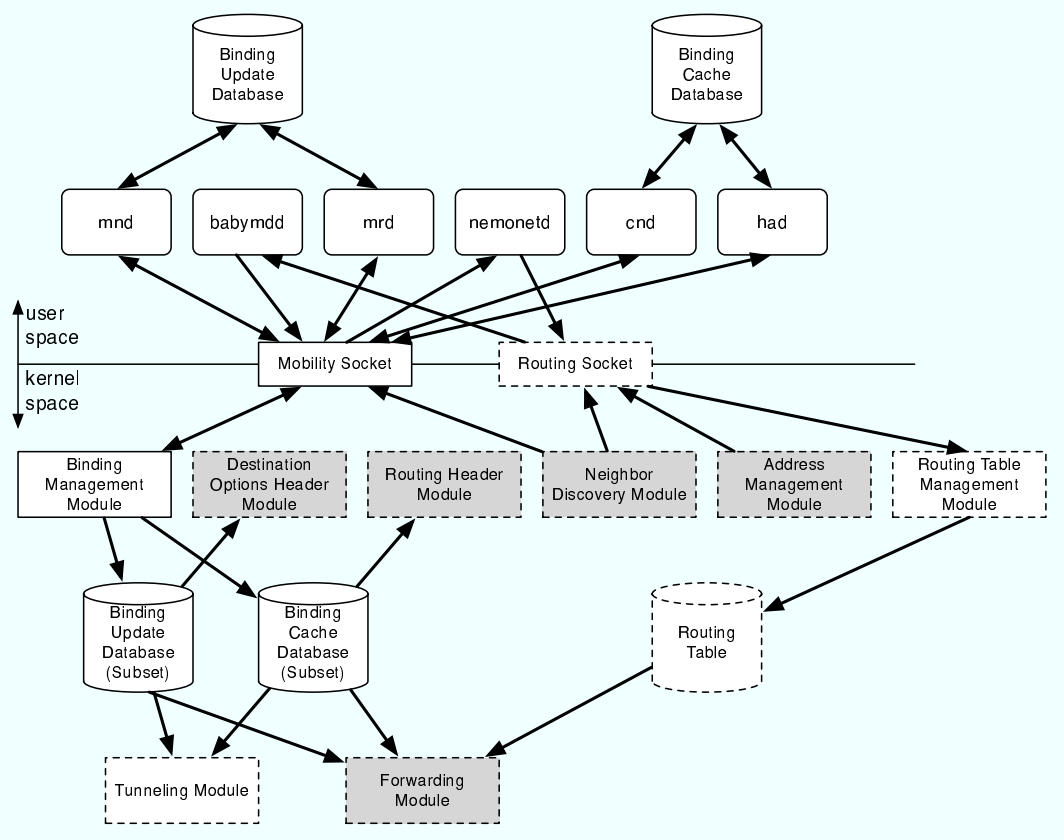

Fig. 6. SHISA Overview

4 Design

4.1 Approaches

This section presents the design principle of our DSMIPv6 implementation. In accordance with the SHISA design principles, we defined the requirements of our DSMIPv6 implementation as follows:

1. Separation of signaling processing and packet routing:
   This is for good performance and efficiency in developing the stack, as explained in Sec. 3.
2. Minimum modification on the existing kernel:
   In order to integrate the implementation to the main branch of BSD operating systems, it is reasonable to minimize the modification on the kernel.
3. Minimum modification on the existing SHISA daemons:
   As the specification reuses the Mobile IPv6 functions for an IPv4 mobility management, it can be expected that many parts of the Mobile IPv6 implementation will be reused.

4.2 Functional Requirements

We defined functional requirements as listed below based on the approaches. The process flowchart of DSMIPv6 is shown in Fig. 7. The circles are functions in the user land space and the square boxes are functions in the kernel space. The requirements are corresponded to each function in the figure.

1. Binding management
   Support IPv4 care-of addresses and IPv4 home addresses. It should be done in the user land programs according to SHISA desgin.
2. Detecting IPv4 care-of addresses
   It should be done in the user land programs same as IPv6 care-of address detection.
3. Sending binding update messages
   Send a binding update message with the IPv4 home address option via IPv6 network, or a

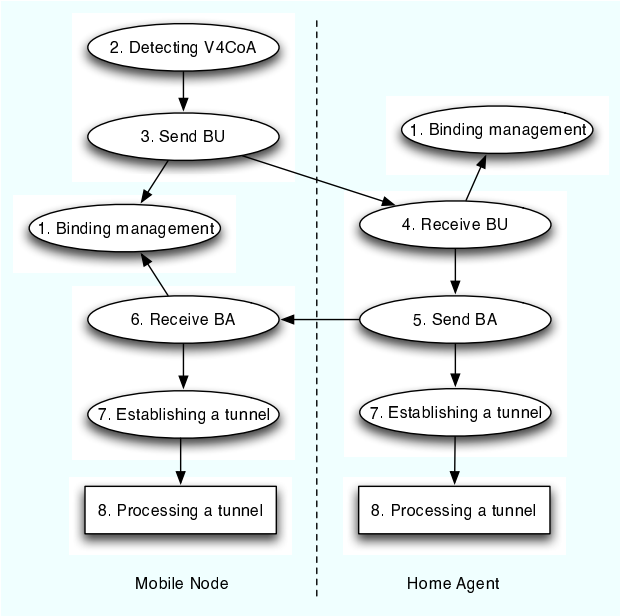

Fig. 7. DSMIPv6 process flowchart

binding update via IPv4 network. It should be done in the user land programs according to SHISA design.

4. Receiving binding update messages
   It should be done in the user land programs according to SHISA design.
5. Receiving binding acknowledgment messages
   Receive a binding acknowledgment message with the IPv4 home acknowledgment option via IPv6 network, or a binding acknowledgment message via IPv4 network. It should be done in the user land programs according to SHISA design.
6. Sending binding acknowledgment messages
   It should be done in the user land programs in the same way as sending binding update messages.
7. Establishing a bi-directional tunnel
   Establish a tunnel between an IPv4 care-of address and an IPv4 home agent address. The control should be done in the user land programs.
8. Processing bi-directional tunnel
   the processing should be done in the kernel, according to SHISA design.

5 Implementation

The SHISA daemons are modified as explained in the following sections to support the DSMIPv6 signaling processing and tunnel setup. Note that the kernel is not modified for IPv4 tunnel processing because it is already existing in IPv6 stack. Since we do not introduce any new deamon for DSMIPv6, the overview of the DSMIPv6 implementation is the same as Fig. 6.

5.1 New Data Structures and Functions

The following data structures (Table 1) and functions (Table 2) are newly defined for the DSMIPv6 extensions.

Table 1. New Data Structures

Name	**Purpose**
IPv4 UDP socket	send/receive IPv4 UDP-encapsulated signaling
IPv4 Raw socket	send/receive IPv4 encapsulated signaling
IPv4 home address (struct in_addr)	added on the binding update list (struct binding_update_list)
IPv4 home address (struct in_addr)	added on the binding cache (struct binding_cache)
IPv4 home address mobility header option	(struct ip6_mh_opt_ipv4_hoa)
IPv4 home address mobility option	added on the Mobility Header option list (struct mip6_mobility_options)

5.2 Binding Management

The DSMIPv6 specification requires to create a binding cache entry and a binding update list entry for each IPv4 and IPv6 home addresses. However, we put an IPv4 home address entry on the binding update list entry and the binding cache entry of the related IPv6 home address, as shown in Table 1, to simplify the implementation. This is consistent with the specification because all parameters in the IPv4 home address binding expect the home address itself are the same as those of the IPv6 home address.

Table 2. New functions

Name	Purpose
nemo_tun4_set()	setup IPv4 tunnel
nemo_tun4_del()	delete IPv4 tunnel
upd4_input_common()	a common (used by both mobile nodes and home agents use) routine for incoming IPv4 UDP packet
udp4sock_open()	open an IPv4 UDP socket
raw4_input_common()	a common routine for incoming IPv4 Raw packet
raw4sock_open()	open an IPv4 Raw socket
v4_sendmessage()	send mobility signaling via IPv4
mnd_get_v4hoa_by _ifindex()	find IPv4 Home Address by an interface index on the MIP interface (where IPv6 home address is stored)
mip6_find_hal_v4()	find IPv4 Home Agent address

An IPv4 care-of address is stored in the binding update list entries or the binding cache entries using the IPv4-mapped IPv6 format. By this way, the same code are reused to maintain bindings. Whenever a care-of address is used, correspondent functions are called according to its address family.

5.3 Mobile Node Modifications

In order to meet requirements defined in Sec. 4.2, the following modifications are needed for mobile node.

Detecting IPv4 Care-of Addresses First, a mobile node have to detect when it configures IPv4 care-of addresses on its interface. One possible way is to modify a DHCP client to have the same function as BABYMDD (notify MND/MRD of a new care-of address via the Mobility Socket). The advantage is easy to control the address configuration function. The mobile node can actively perform adding or deleting a care-of address depending on the link status. However, this approach makes the comparison procedure between an IPv4 care-of address and an IPv6 care-of address difficult as following. When both IPv4 access and IPv6 access are available on a link, IPv6 care-of address must be chosen as the primary care-of address of the node. If the care-of address detections are separately performed for IPv4 and IPv6, MND/MRD will have the address change notification asynchronously. Therefore, it is reasonable to modify BABYMDD to deal with IPv4 care-of address too.

The overview of this idea is shown in Fig. 8. The circles in the figure represent functions related to the DSMIPv6 operation, and the bold squares represent a part modified especially for the DSMIPv6 implementation.

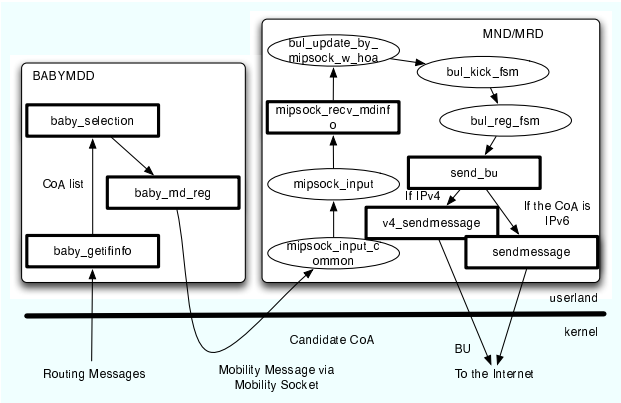

Fig. 8. Detecting IPv4 care-of address and Sending a binding update message

When an IPv4 care-of address is assigned or deleted, a routing message such as RTM_NEWADDR, RTM_DELADDR or RTM_ADDRINFO can be received by monitoring the routing socket. BABYMDD monitors the routing messages to make a list of candidate IPv6 care-of addresses in baby_getifinfo(). We use the same approach for IPv4. A list of candidate IPv4 care-of addresses is also built in baby_getifinfo().

In the BSD operating systems, the *sysctl* allows to retrieve kernel state. The state to be retrieved or set is described using a Management Information Base (MIB) name style. In order to obtain all addresses assigned on a mobile router, baby_getifinfo() uses a *sysctl* with the following MIB.

```
mib[0] = CTL_NET;
mib[1] = PF_ROUTE;
```

```
mib[2] = 0;
mib[3] = AF_UNSPEC;
mib[4] = NET_RT_IFLIST;
mib[5] = 0;
```

BABYMDD then selects a primary care-of address from the list. The function to determine the primary care-of address, baby_selection(), is modified to make an IPv4 care-of address as the primary care-of address. The interface which has the smallest interface index becomes the primary, and an IPv6 care-of address has priority over an IPv4 care-of address as described in [3].

BABYMDD notifies MND/MRD of the selected care-of address via the Mobility Socket. The function to notify MND/MRD of the selected care-of address via the Mobility Socket, baby_md_reg(), is modified to carry both IPv6 and IPv4 addresses. The data structure to store the care-of address was changed from *struct sockaddr_in6* to *struct sockaddr_storage*. Whenever used, the structure is casted with *struct sockaddr_in* or *sockaddr_in6* according to its address family.

Sending a Binding Update Message The notification is passed though generic mipsock processing routines such as mipsock_input_common(), mipsock_input() and mipsock_recv_mdinfo() as shown in Fig. 8. MND/MRD then updates its binding update list with the primary care-of address by bul_update_by_mipsock_w_hoa(). The binding update list is processed by SHISA binding state machine with bul_kick_fsm() and bul_reg_fsm(). A binding update message is sent by calling send_bu() if everything is successfully processed.

The function to receive Mobility Socket messages, mipsock_recv_mdinfo(), is modified to receive an IPv4 care-of address. If the care-of address is IPv4, it will be encoded into an IPv4-mapped IPv6 address. Thanks to this operation, it is not required to modify all other binding management functions.

The function to send a BU, send_bu(), is modified to include the IPv4 home address mobility option. If the care-of address is an IPv4 address, Alternate Care-of Address option is not added but IPv4 home address mobility option is added. If the care-of address is an IPv4 address, v4_sendmessage() is called.

According to [3], the BU is encapsulated in an IPv4 UDP packet. Thus, a new function to send an IPv4 message, v4_sendmessage(), is needed to send an IPv4 message.

IPv4 tunnel packets are decapsulated by the forwarding module implemented in the kernel (shown in Fig. 6), and a pair of a care-of address and a home agent address is to be referred by the module for the decapsulation. Therefore, a tricky hack to setup an IPv4 tunnel (store the pair into the kernel) at the same time as sending a BU is implemented in order to receive the correspondent BA.

Receiving a Binding Acknowledgment Message If the care-of address is IPv4, the binding acknowledgement message is encapsulated in whether IPv4 or IPv4 UDP as explained in Sec. 5.4. The mobile node' operation after receiving an binding acknowledgment message is shown in Fig. 9.

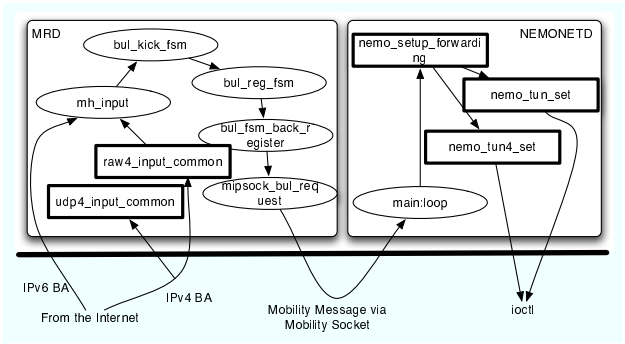

Fig. 9. Receiving a BA and Establishing a Bi-directional Tunnel

When MND/MRD receives an IPv4 or IPv4 UDP encapsulated binding acknowledgment message, the packet will be processed in raw4_input_common(), or in udp4_input_common() in the case of UDP.

The received IPv4/IPv4 UDP packet goes through sanity checks. The sanity checks consist of verifying whether the packet contains an IPv6 home address option, and whether the source address in the IPv4 header is the same as the source

address in the IPv6 header (in a IPv6-mapped IPv4 address format). If they are not the same, it is assumed that a NAT is existed between the mobile node and the home agent. After the checks, the packet will be decapsulated and carried to mh_input().

The acknowledgment is then passed to the SHISA mobility header processing routine, mh_input(). The status of the binding is updated as [the binding is accepted] in bul_kick_fsm(), bul_reg_fsm(), and bul_fsm_back_register().

Establishing a Bi-directional Tunnel If a mobile node is a mobile router, a request to create a bi-directional tunnel between the home agent and the mobile router is sent via the Mobility Socket to NEMONETD by mipsock_bul_request() after the binding information has been registered.

NEMONETD receives the request at its main loop, and the bi-directional tunnel is established by calling nemo_setup_forwarding() and nemo_tun4_set().

The function to setup a bi-directional IPv6-in-IPv6 tunnel, nemo_tun_setup(), is modified to establish an IPv4 tunnel. Since the tunnel is already established before sending the BU, most parts of this function will be skipped if the care-of address is an IPv4 address.

The function to setup IPv4 tunnel, nemo_tun4_set(), is added. The function to delete IPv4 tunnel, nemo_tun4_del(), is also added. Those functions are called according to the address family of the primary care-of address.

5.4 Home Agent Modifications

As defined in Sec. 4.2, the following modifications are needed on a Home Agent: Receiving Binding Update Message, Setup a bi-directional tunnel and Sending Binding Acknowledgment Message. Fig. 10 shows the details.

Receiving a Binding Update Message When HAD receives an IPv4 UDP encapsulated binding update message, it will be processed in upd4_input_common(). The received IPv4 UDP packet goes through sanity checks defined in [3].

The binding update message is passed to the SHISA mobility header processing routine, mh_input(), and a function to process the binding update, receive_bu(), is called. If the binding update message is processed correctly, the binding cache entry is created or updated by mip6_bc_add() and a request to create a bi-directional tunnel between the mobile router and the home agent is sent via the Mobility Socket to NEMONETD by misock_bc_request().

The function to process a binding update message, receive_bu(), is modified to accommodate IPv4-mapped IPv6 address in the care-of address field. The function to process/expand mobility options, get_mobility_options(), is also modified to process the IPv4 home address mobility option.

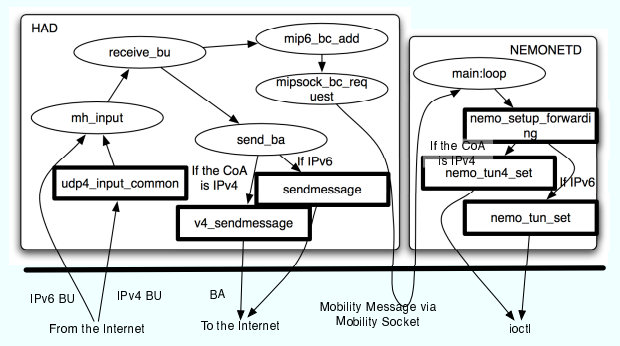

Fig. 10. Receiving a binding update message, Establishing a bi-directional tunnel, and Sending a binding acknowledgement message

Sending a Binding Acknowledgment Message The binding acknowledgment message is then sent by send_ba() as shown in Fig. 10. The function to send a binding acknowledgment message, send_ba(), is modified to add the IPv4 acknowledgment option. If the care-of address is an IPv4 address, the IPv4 address acknowledgment option is added. The binding acknowledgment message is sent via IPv4 network by v4_sendmessage(). The v4_sendmessage() is a function for IPv4 encapsulation or IPv4 UDP encapsulation.

Establishing a Bi-directional Tunnel If the home agent is capable of NEMO Basic Support, a request to create a bi-directional tunnel is sent

via the Mobility Socket to NEMONETD by mipsock_bul_request() after the binding registered as shown in Fig. 10.

NEMONETD receives the request at its main loop, and the bi-directional tunnel is established by calling nemo_setup_forwarding() as well as the mobile node's operation.

5.5 Initialization

The main() in HAD is modified to initialize new sockets. The main() in MND/MRD is also modified as follows. Users can specify an IPv4 home agent address as one of the MND/MRD's argument. This is because the Dynamic Home Agent Address Discovery is not defined in the specification [3]. New sockets are initialized here as well.

The function to build the home agent list from MND/MRD's arguments, add_hal_by_commandline_xxx(), is modified to add IPv4 home agent addresses too. As both IPv4 and IPv6 home agent addresses are stored into the same list, appropriate home agent address is chosen by confirming its address family (the address family must be the same as the current primary care-of address). A new function, mnd_get_v4hoa_by_ifindex() which gets an IPv4 home address from mip device is added.

The function to make a list of home addresses, hoainfo_insert(), is added to add an IPv4 home address on the list. There is an design assumption that an IPv4 home address is assigned on the same interface as where an IPv6 home address is assigned.

6 Evaluation

We performed experiments using our DSMIPv6 implementation in order to measure signalling processing costs newly introduced by DSMIPv6 and to confirm that our implementation works as expected.

6.1 Testbed Details

In order to evaluate our DSMIPv6 implementation, we have setup a testbed as shown in Fig. 11.

A mobile router (MR) is a laptop, IBM ThinkPad X31, with Intel Pentium M 1298.97MHz, 512MB memory, Intel Pro/1000 VE Network Controller, BUFFALO WLI2-CF-S11 (IEEE802.11b), and an EvDO 1x (2GHz) cellular card . SHISA on NetBSD 2.0 is running on this router.

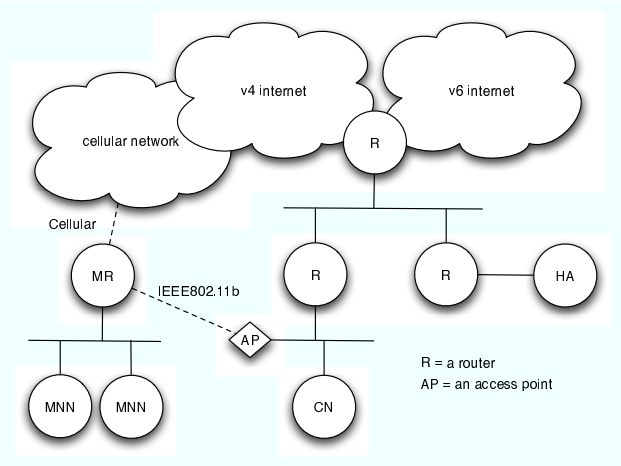

Fig. 11. Testbed topology

A home agent (HA) is a 1U server with Intel Pentium 2527.16MHz, 256MB memory, and Sundance ST-201 10/100 Ethernet. SHISA on NetBSD 2.0 is running on this server.

6.2 Signaling Processing Costs

The DSMIPv6 specification introduces the IPv4 care-of address detection and the additional IPv4 headers. The processing costs are listed as follows and measured by using the testbed.

1. Detecting a care-of address
   a time from when the mobile router attached to the link, to when baby_getifinfo() is called.
2. Sending a binding update
   a time from when baby_getifinfo() is called to when v4_sendmessage() is called
3. Receiving a binding update
   a time from when udp4_input_common() is called to when receive_bu() is called.
4. Sending a binding acknowledgment
   a time from when receive_bu() is called to when v4_sendmessage() is called.
5. Receiving a binding acknowledgment
   from when raw4_input_common() to when bul_kick_ fsm() is called.

The scenario is that an address configuration function (DHCP or NDP) is launched when the mobile node attaches to the Wireless LAN link and the above operations are then performed. This experiments are performed 200 times for the case that the mobile node attaches to IPv6 global foreign network (MIPv6 is used) and the case of IPv4 global foreign network (DSMIPv6 is used). The results are shown in Table 3.

Table 3. Signaling Costs (msec)

Item	1	2	3	Item	4	5
MIPv6	819.077	1.612	0.232	MIPv6	1.101	0.234
DSMIPv6	1818.758	2.351	0.268	DSMIPv6	1.140	0.316

The time to detect the care-of address are largely differ. This is because DHCP requires two round trips between the mobile node and the access router whereas NDP requires only one round trip.

As expected, additional costs due to the outer IPv4 header are observed. However, these costs are negligible because they are smaller than the round trip between the mobile node and the home agent, and these processing are not happened so often (for example, the default lifetime of bindings in SHISA is 40 seconds).

6.3 Forwarding Operation Checks and its Performances

The evaluation of the forwarding module is conducted for all cases (IPv4 or IPv6 home addresses v.s. IPv4 or IPv6 care-of addresses) by using the EvDO 1x 2GHz cellular with the experimental access point at YRP Research Center, KDDI R&D Laboratories. The experimental access point can be configured as an IPv6 global foreign network or an IPv4 global foreign network. This environment is less interference than the commercial access points because there is no user other than us.

The results are listed in Table 4. The first column represents the test case, whether the care-of address (CoA) is IPv6 or IPv4 and whether the correspondent node (CN) uses IPv6 or IPv4. The round trip time (RTT) is measured by using *ping* and the throughput is measured by using *iperf*.

Table 4. Performance in All Situations

Case CoA-CN	RTT (ping) (msec)	Throughput (bps) TCP (up/down)	Throughput (bps) UDP (up/down)
v6-v6	174.787	87K/238K	95.3K/332K
v6-v4	183.6	104.3K/701K	96.3K/344.4K
v4-v6	149.8	112K/1.05M	111K/324K
v4-v4	183.27	103.2K/1.08M	110K/308.6K

It is confirmed that the forwarding functions works in all situations. In general, the header cost reduction effect is expected when IPv4 is used because the IPv4 header (20 bytes) is smaller than the IPv6 header (40 bytes). However, it is not observed in this experiment. The performance thus might be greatly depended on other elements such as the round trip time, MTU, network congestion, and the packet loss.

6.4 Consideration

The DSMIPv6 protocol works in all situation without adding remarkable header processing costs. Since only 874 lines are added on the SHISA user land programs for the DSMIPv6 support , it can be managed to minimize the modification (SHISA has 20787 lines). However, the following issues are needed to be revised in the spec:

- UDP header in a binding acknowledgment
  In order to receive a IPv4 encapsulated binding acknowledgment, an IPv4 tunnel is setup before accepting the binding acknowledgment. This can become a security hole. To avoid this, we suggest to force a UDP header in the binding acknowledgment always. (This suggestion has been adopted in the IETF working group and included in the next version of the specification).
- Uses of the IPv4-mapped IPv6 address
  The mapped address in an IPv6 header is not preferable because of several reasons [8].

Thus, we suggest to put the IPv6 home address in the IPv6 header and a kind of IPv4 care-of address option in the mobility headers.

7 Conclusion

This paper describes the DSMIPv6 implementation on an open source Mobile IPv6 and NEMO implementation for BSD operating systems. The DSMIPv6 implementation was designed to separate the signaling function and the forwarding function, and to minimize modifications on the existing kernel and user land programs. The binding management code is shared with both IPv4 and IPv6 mobility functions by using IPv4-mapped IPv6 address format. When care-of or home addresses are used, new or modified correspondent functions are called according to its address family. An IPv4 care-of address detection feature is also added to the user land programs. By evaluating the implementation, it is confirmed that the DSMIPv6 implementation works in the all situations without adding remarkable header processing overhead. It is thus said that the specification is stable to forward IPv4/IPv6 packets address to their home addresses/mobile networks.

Acknowledgment

The authors would like to thank KDDI R&D Laboratories and KDDI for their helps.

References

1. D. Johnson et al. Mobility Support in IPv6. Request For Comments 3775, The Internet Engineering Task Force, June 2004.
2. V. Devarapalli et al. Network Mobility (NEMO) Basic Support Protocol. Request For Comments 3963, The Internet Engineering Task Force, Jan. 2005.
3. H. Soliman Ed. Mobile IPv6 support for dual stack Hosts and Routers (DSMIPv6). Internet Drafts draft-ietf-mip6-nemo-v4traversal-02, The Internet Engineering Task Force, June 2006.
4. SHISA Project, As of July. 2006. http://www.mobileip.jp/.
5. Keiichi Shima, Ryuji Wakikawa, Koshiro Mitsuya, Keisuke Uehara, and Tsuyoshi Momose. SHISA: The IPv6 Mobility Framework for BSD Operating Systems. In *IPv6 Today – Technology and Deployment Workshop (IPv6TD'06)*, August 2006.
6. KAME Project, As of July. 2006. http://www.kame.net/.
7. T. Momose et al. The application interface to exchange mobility information with Mobility subsystem. Internet Drafts draft-momose-mip6-mipsock-00, The Internet Engineering Task Force, July 2005.
8. Craig Metz et al. IPv4-Mapped Address API Considered Harmful. Internet Drafts draft-cmetz-v6ops-v4mapped-api-harmful-01, The Internet Engineering Task Force, October 2003.