Samita Chakrabarti
Erik Nordmark
Sun Microsystems, Inc.
October, 2003

Extension to Sockets API for Mobile IPv6

<draft-chakrabarti-mobileip-mipext-advapi-02.txt>

Status of this Memo

Copyright Notice

Abstract

This document describes data structures and API support for Mobile IPv6 as an extension to Advanced Socket API support for IPv6.

Mobility Support in IPv6 introduces mobility protocol header for IPv6. It is expected that future Mobile IPv6 applications and implementations may need to access Mobility binding messages and Return Routability messages for diagnostic, packet accounting and local policy setting purposes. In order to provide portability for Mobile IP applications that use sockets under IPv6, standardization is needed for the Mobile IPv6 specific APIs.

This document provides mechanism for API access to retrieve and set information for Mobility Header messages, Home address destination options and Type 2 Routing header extension headers. It also discusses the common data structures and definitions that might be used by advanced Mobile IPv6 socket applications.

Table of Contents

1. Introduction

Mobility Support in IPv6 [2] defines a new mobility protocol header, home address destination option and a new routing header type. It is expected that Mobile IPv6 user-level implementations and some applications will need to access and process these IPv6 extension headers. This document is an extension to existing Advanced Sockets API document [1]; it addresses the IPv6 Sockets API for Mobile IPv6 protocol support. The target applications for this socket API is believed to be the debugging and diagnostic applications as well as some policy applications which would like to receive a copy of protocol information at the application layer.

This document can be divided into the following parts.

1. Definitions of constants and structures for C programs that capture the Mobile IPv6 packet formats on the wire. A common definition of these is useful at least for packet snooping appplications. This is captured in section 2.

2. Notes on how to use the IPv6 Advanced API to access home address options and type 2 routing headers. This is captured in section 3.

3. Notes on how user-level applications can observe MH (Mobility Header) packets using raw sockets (in section 4). The IPv6 RAW socket interface described in this document, allows applications to receive MH packets whether or not the systems MH processing takes place in the "kernel" or at the "user space".

4. Suggested name for /etc/protocols (in section 5).

It is anticipated that Mobile IPv6 will be used widely from mobile devices to Server and Routing platforms. Thus it is useful to have a standard API for portability of Mobile IPv6 applications on a wide variety of platforms and operating systems.

The packet information along with access to the extension headers (Routing header and Destination options) are specified using the "ancillary data" fields that were added to the 4.3BSD Reno sockets API in 1990. The reason is that these ancillary data fields are part of the Posix.1g standard and should therefore be adopted by most vendors. This is in conformance with Advanced API for IPv6 sockets [1].

This document does not address application access to either the authentication header or the encapsulating security payload header.

All examples in this document omit error checking in the favor of brevity.

We note that many of the functions and socket options defined in this document may have error returns that are not defined in this document. Many of these possible error returns will be recognized only as implementations proceed.

Datatypes in this document follow the Posix.1g format: intN_t means a signed integer of exactly N bits (e.g., int16_t) and uintN_t means an unsigned integer of exactly N bits (e.g., uint32_t).

This document provides guidelines on Mobile IPv6 socket applications and believes that some other appropriate standardization body will standardize the APIs along with other IPv6 advanced socket APIs.

2. Common Structures and Definitions

This API assumes that the fields in the protocol headers are left in the network byte order, which is big-endian for the Internet protocols. If not, then either these constants or the fields being tested must be converted at run-time, using something like htons() or htonl().

A new header file : <netinet/ip6mh.h>

2.1. The Mobility Header Data Structures

2.1.1 The ip6_mh Structure

The following structure is defined as a result of including <netinet/ip6mh.h>. This is fixed part of the mobility header.

```
struct  ip6_mh {
     uint8_t    ip6mh_proto;   /* NO_NXTHDR by default */
     uint8_t    ip6mh_hdrlen;  /* Header Len in unit of 8 Octets
                                  excluding the first 8 Octets */
     uint8_t    ip6mh_type;    /* Type of Mobility Header */
     uint8_t    ip6mh_reserved;   /* Reserved */
     uint16_t   ip6mh_cksum;  /* Mobility Header Checksum */
     /* Followed by type specific messages */
};
```

2.1.2 Binding Refresh Request Mobility Message

```
  struct  ip6_mh_binding_request {
     struct ip6_mh  ip6mhbr_hdr;
     uint16_t   ip6mhbr_reserved;
     /* Followed by optional Mobility Options */
  };
```

2.1.3 Home Address Test Init (HoTI) Message

```
   struct   ip6_mh_home_test_init {
     struct  ip6_mh  ip6mhhti_hdr;
     uint16_t        ip6mhhti_reserved;
     uint32_t        ip6mhhti_cookie[2]; /* 64 bit Cookie by MN */
     /* Followed by optional Mobility Options */
   };
```

2.1.4 Care-of Address Test Init (CoTI) Message

```
      struct   ip6_mh_careof_test_init {
        struct  ip6_mh  ip6mhcti_hdr;
        uint16_t        ip6mhcti_reserved;
        uint32_t        ip6mhcti_cookie[2]; /* 64 bit Cookie by MN */
        /* Followed by optional Mobility Options */
      };
```

2.1.5 Home Address Test (HOT) Message

```
     struct  ip6_mh_home_test {
        struct  ip6_mh  ip6mht_hdr;
        uint16_t    ip6mhht_nonce_index;
        uint32_t    ip6mhht_cookie[2];    /* Cookie from HOTI msg */
        uint32_t    ip6mhht_keygen[2];  /* 64 Bit Key by CN */
        /* Followed by optional Mobility Options */
    };
```

2.1.6 Care Of Address Test (COT) Message

```
     struct  ip6_mh_careof_test {
        struct  ip6_mh  ip6mhct_hdr;
        uint16_t   ip6mhct_nonce_index;
        uint32_t   ip6mhct_cookie[2]; /* Cookie from COTI message */
        uint32_t   ip6mhct_keygen[2];  /* 64bit key by CN */
        /* Followed by optional Mobility Options */
      };
```

2.1.7 Binding Update Mobility Message

```
    struct ip6_mh_binding_update {
       struct ip6_mh ip6mhbu_hdr;
       uint16_t    ip6mhbu_seqno;      /* Sequence Number */
       uint16_t    ip6mhbu_flags;
       uint16_t    ip6mhbu_lifetime; /* Time in unit of 4 sec */
        /* Followed by optional Mobility Options */
    };

    /* Binding Update Flags, in network byte-order */
    #define IP6_MH_BU_ACK    0x8000  /* Request a binding ack */
    #define IP6_MH_BU_HOME   0x4000  /* Home Registration */
    #define IP6_MH_BU_LLOCAL 0x2000  /* Link-local compatibility */
    #define IP6_MH_BU_KEYM   0x1000  /* Key management mobility  */
```

2.1.8 Binding Acknowledgment Mobility Message

```
     struct  ip6_mh_binding_ack {
        struct ip6_mh  ip6mhba_hdr;
        uint8_t   ip6mhba_status;    /* Status code */
        uint8_t   ip6mhba_flags;
        uint16_t  ip6mhba_seqno;
        uint16_t  ip6mhba_lifetime;
        /* Followed by optional Mobility Options */
     };


    /* Binding Acknowledgement Flags */
    #define IP6_MH_BA_KEYM       0x80    /* Key management mobility */
```

2.1.9 Binding Error Mobility Message

```
     struct   ip6_mh_binding_error {
        struct ip6_mh   ip6mhbe_hdr;
        uint8_t        ip6mhbe_status;  /* Error Status */
        uint8_t        ip6mhbe_reserved;
        struct in6_addr ip6mhbe_homeaddr;
        /* Followed by optional Mobility Options */
      };
```

2.1.10 Mobility Option TLV data structure

```
       struct   ip6_mh_opt {
          uint8_t    ip6mhopt_type;   /* Option Type */
          uint8_t    ip6mhopt_len;    /* Option Length */
          /* Followed by variable length Option Data in bytes */
       };
```

2.1.11 Mobility Option Data Structures

2.1.11.1 Binding Refresh Advice

```
         struct ip6_mh_opt_refresh_advice {
           uint8_t  ip6mora_type;
           uint8_t  ip6mora_len;
           uint16_t ip6mora_interval; /* Refresh interval in 4 sec */
        };
```

2.1.11.2 Alternate Care-of Address

```
        struct ip6_mh_opt_altcoa {
           uint8_t ip6moa_type;
           uint8_t ip6moa_len;
           uint8_t ip6moa_addr[16];   /* Alternate Care-of Address */
        };
```

2.1.11.3 Nonce Indices

```
         struct ip6_mh_opt_nonce_index {
            uint8_t ip6moni_type;
            uint8_t ip6moni_len;
            uint16_t ip6moni_home_nonce;
            uint16_t ip6moni_coa_nonce;
         };
```

2.1.11.4 Binding Authorization Data

```
         struct ip6_mh_opt_auth_data {
            uint8_t ip6moad_type;
            uint8_t ip6moad_len;
            /* Followed by authentication data */
         };
```

2.2 Mobility Header Constants

```
       IPv6 Next Header Value for Mobility:
       <netinet/in.h>

       #define IPPROTO_MH       62 /* IPv6 Mobility Header: IANA-TBD */
```

Mobility Header Message Types:
<netinet/ip6mh.h>

```
#define IP6_MH_TYPE_BRR       0   /* Binding Refresh Request */
#define IP6_MH_TYPE_HOTI      1   /* HOTI Message   */
#define IP6_MH_TYPE_COTI      2   /* COTI Message  */
#define IP6_MH_TYPE_HOT       3   /* HOT Message   */
#define IP6_MH_TYPE_COT       4   /* COT Message  */
#define IP6_MH_TYPE_BU        5   /* Binding Update */
#define IP6_MH_TYPE_BACK      6   /* Binding ACK */
#define IP6_MH_TYPE_BERROR    7   /* Binding Error */
```

Mobility Header Message Option Types:
<netinet/ip6mh.h>

```
#define  IP6_MHOPT_PAD1       0x00  /* PAD1 */
#define  IP6_MHOPT_PADN       0x01  /* PADN */
#define  IP6_MHOPT_BREFRESH   0x02  /* Binding Refresh */
#define  IP6_MHOPT_ALTCOA     0x03  /* Alternate COA */
#define  IP6_MHOPT_NONCEID    0x04  /* Nonce Index */
#define  IP6_MHOPT_BAUTH      0x05  /* Binding Auth Data */
```

Status values accompanied with Mobility Binding Acknowledgement:
<netinet/ip6mh.h>

```
#define IP6_MH_BAS_ACCEPTED          0   /* BU accepted */
#define IP6_MH_BAS_PRFX_DISCOV       1   /* Accepted, but prefix
                                            discovery Required */
#define IP6_MH_BAS_UNSPECIFIED       128 /* Reason unspecified */
#define IP6_MH_BAS_PROHIBIT          129 /* Administratively
                                              prohibited */
#define IP6_MH_BAS_INSUFFICIENT      130 /* Insufficient
                                              resources */
#define IP6_MH_BAS_HA_NOT_SUPPORTED  131 /* HA registration not
                                              supported */
#define IP6_MH_BAS_NOT_HOME_SUBNET   132  /* Not Home subnet */
#define IP6_MH_BAS_NOT_HA            133  /* Not HA for this
                                               mobile node */
#define IP6_MH_BAS_DAD_FAILED        134  /* DAD failed */
#define IP6_MH_BAS_SEQNO_BAD         135  /* Sequence number out
                                               of range */
```

```
 #define IP6_MH_BAS_HOME_NI_EXPIRED   136  /* Expired Home nonce
                                                index */
 #define IP6_MH_BAS_COA_NI_EXPIRED    137  /* Expired Care-of
                                                nonce index */
 #define IP6_MH_BAS_NI_EXPIRED        138  /* Expired Nonce
                                                Indices */
 #define IP6_MH_BAS_REG_NOT_ALLOWED   139  /* Registration type
                                                change disallowed */
```

Status values for the Binding Error mobility messages:
<netinet/ip6mh.h>

```
#define IP6_MH_BES_UNKNOWN_HAO    1 /* Unknown binding for HOA */
#define IP6_MH_BES_UNKNOWN_MH     2 /* Unknown MH Type */
```

2.3. IPv6 Home Address Destination Option

```
<netinet/ip6.h>

/* Home Address Destination Option */
struct ip6_opt_home_address {
   uint8_t           ip6oha_type;
   uint8_t           ip6oha_len;
   uint8_t           ip6oha_addr[16];   /* Home Address */
};
```

Option Type Definition:

```
#define IP6OPT_HOME_ADDRESS             0xc9    /* 11 0 01001 */
```

2.4 Type 2 Routing Header

```
<netinet/ip6.h>

 /* Type 2 Routing header for Mobile IPv6 */
  struct ip6_rthdr2 {
    uint8_t  ip6r2_nxt;       /* next header */
    uint8_t  ip6r2_len;       /* length : always 2 */
    uint8_t  ip6r2_type;      /* always 2 */
    uint8_t  ip6r2_segleft;   /* segments left: always 1 */
    uint32_t ip6r2_reserved;  /* reserved field */
    struct in6_addr ip6r2_homeaddr;  /* Home Address */
  };
```

2.5 New ICMP Messages for Mobile IPv6

ICMP message types and definitions for Mobile IPv6 are defined in <netinet/icmp6.h>

```
#define MIP_HA_DISCOVERY_REQUEST    150
#define MIP_HA_DISCOVERY_REPLY      151
#define MIP_PREFIX_SOLICIT          152
#define MIP_PREFIX_ADVERT           153
```

The following data structures can be used for the ICMP message types discussed in section 6.5 through 6.8 in the base Mobile IPv6 [2] specification.

```
struct mip_dhaad_req {          /* Dynamic HA Address Discovery */
       struct  icmp6_hdr   mip_dhreq_hdr;
};

#define  mip_dhreq_type      mip_dhreq_hdr.icmp6_type
#define  mip_dhreq_code      mip_dhreq_hdr.icmp6_code
#define  mip_dhreq_cksum     mip_dhreq_hdr.icmp6_cksum
#define  mip_dhreq_id        mip_dhreq_hdr.icmp6_data16[0]
#define  mip_dhreq_reserved  mip_dhreq_hdr.icmp6_data16[1]


struct mip_dhaad_rep {         /* HA Address Discovery Reply */
       struct icmp6_hdr   mip_dhrep_hdr;
       /* Followed by Home Agent IPv6 addresses */
};

#define  mip_dhrep_type      mip_dhrep_hdr.icmp6_type
#define  mip_dhrep_code      mip_dhrep_hdr.icmp6_code
#define  mip_dhrep_cksum     mip_dhrep_hdr.icmp6_cksum
#define  mip_dhrep_id        mip_dhrep_hdr.icmp6_data16[0]
#define  mip_dhrep_reserved  mip_dhrep_hdr.icmp6_data16[1]


struct mip_prefix_solicit {   /* Mobile Prefix Solicitation */
       struct icmp6_hdr     mip_ps_hdr;
};

#define  mip_ps_type          mip_ps_hdr.icmp6_type
#define  mip_ps_code          mip_ps_hdr.icmp6_code
#define  mip_ps_cksum         mip_ps_hdr.icmp6_cksum
#define  mip_ps_id            mip_ps_hdr.icmp6_data16[0]
#define  mip_ps_reserved      mip_ps_hdr.icmp6_data16[1]
```

```
struct mip_prefix_advert {    /* Mobile Prefix Adverisements */
       struct  icmp6_hdr   mip_pa_hdr;
        /* Followed by one or more PI options */
};

#define  mip_pa_type            mip_pa_hdr.icmp6_type
#define  mip_pa_code            mip_pa_hdr.icmp6_code
#define  mip_pa_cksum           mip_pa_hdr.icmp6_cksum
#define  mip_pa_id              mip_pa_hdr.icmp6_data16[0]
#define  mip_pa_flags_reserved  mip_pa_hdr.icmp6_data16[1]

#define  MIP_PA_FLAG_MANAGED    0x8000
#define  MIP_PA_FLAG_OTHER      0x4000
```

Prefix options are defined in IPv6 Advanced Socket API [1].
Mobile IPv6 Base specification [2] describes the modified
behavior in 'Modifications to IPv6 Neighbor Discovery' section.
Prefix Options for Mobile IP are defined in the following section.

2.6 IPv6 Neighbor Discovery Changes

IPv6 Neighbor Discovery changes are also defined in
<netinet/icmp6.h>

New 'Home Agent' flag in router advertisement:

```
#define  ND_RA_FLAG_HOMEAGENT   0x20  /* Home Agent flag in RA */
```

New Router flag with prefix information of the home agent:

```
#define  ND_OPT_PI_FLAG_ROUTER  0x20  /* Router flag in PI */
```

As per Mobile IPv6 specification [2] a Home Agent MUST include
at least one prefix option with the Rouer Address (R) bit set.
Advanced Socket API [1] defines data structure for prefix option
as follows:

```
struct nd_opt_prefix_info {    /* prefix information */
   uint8_t   nd_opt_pi_type;
   uint8_t   nd_opt_pi_len;
   uint8_t   nd_opt_pi_prefix_len;
   uint8_t   nd_opt_pi_flags_reserved;
   uint32_t  nd_opt_pi_valid_time;
   uint32_t  nd_opt_pi_preferred_time;
   uint32_t  nd_opt_pi_reserved2;
   struct in6_addr  nd_opt_pi_prefix;
 };
```

New advertisement interval option and home agent information options are defined in Mobile IPv6 [2] base specification.

```
struct nd_opt_adv_interval {     /* Advertisement interval option */
   uint8_t        nd_opt_ai_type;
   uint8_t        nd_opt_ai_len;
   uint16_t       nd_opt_ai_reserved;
   uint32_t       nd_opt_ai_interval;
};
```

The option types for the new Mobile IPv6 specific options:

```
#define  ND_OPT_ADV_INTERVAL    7     /* Adv Interval Option  */
#define  ND_OPT_HA_INFORMATION  8     /* HA Information option */

struct nd_opt_homeagent_info {  /* Home Agent information */
   uint8_t        nd_opt_hai_type;
   uint8_t        nd_opt_hai_len;
   uint16_t       nd_opt_hai_reserved;
   uint16_t       nd_opt_hai_preference;
   uint16_t       nd_opt_hai_lifetime;
};
```

3. Access to Home Address Destination Option and Routing Headers

Applications that need to be able to access home address destination option and routing header type 2 information should use the same mechanism defined in Advanced Sockets API for IPv6 in section 4.

In order to receive Home Address destination option or route header type 2 extension header, application must call setsockopt() to turn on the corresponding flag:

```
int  on = 1;

setsockopt(fd, IPPROTO_IPV6, IPV6_RECVRTHDR,    &on, sizeof(on));
setsockopt(fd, IPPROTO_IPV6, IPV6_RECVDSTOPTS,  &on, sizeof(on));
```

When any of these options are enabled, the corresponding data is returned as control information by recvmsg(), as one or more ancillary data objects. Receiving the above information for TCP applications is not defined in this document (see section 4.1 of Advanced Sockets API for IPv6[1].

For sending home address destination option, ancillary data can be used to specify the option content for a single datagram. This only applies to datagram and raw sockets; not to TCP sockets. Advanced API [1] document restricts one IPV6_xxx ancillary data object for a particular extension header in the control buffer. Thus there would be a single ancillary data object for Home address destination option in a ancillary data buffer. If the kernel implementation supports this API, it is responsible for extracting the Home address destination option data object and placing it as destination option extension header in compliance with section 6.3 of Mobile IPv6 [2] base specification.

For TCP data packets with home-address destination option may be used with "sticky" option for all transmitted packets. However, at this point, it is unknown why an application would want to set home-address option or Route Header Type 2 extension header along with its data packets as Mobile IPv6 protocol takes care of them transparently at the protocol stack.

However, the following socket option parameters and cmsghdr fields may be used for sending.

```
opt level/    optname/          optval/
cmsg_level    cmsg_type         cmsg_data[]
------------  ------------      ------------------------
IPPROTO_IPV6  IPV6_DSTOPTS      ip6_dest structure
IPPROTO_IPV6  IPV6_RTHDR        ip6_rthdr structure
```

Some IPv6 implementations may support "sticky" options [1] for IPv6 destination option for datagram sockets.

3.1 Routing Header access functions

While accessing Routing header Type 2 extension header, one MUST use type = 2 and segment = 1. The following functions are supported for Mobile IPv6 applications for sending and receiving Routing Header Type 2 headers:

```
size_t inet6_rth_space(int type, int segments);
void *inet6_rth_init(void *bp, int bp_len, int type, int segments);
int inet6_rth_add(void *bp, const struct in6_addr *addr);
int inet6_rth_segments(const void *bp);
struct in6_addr *inet6_rth_getaddr(const void *bp, int index);
```

NOTE: Reversing operation is not possible using Route Header Type 2 extension header.

Detail description and examples of accessing a IPv6 Routing Header are discussed in Advanced API for IPv6 Sockets [1].

3.2 Home Address Destination Option access functions

The application must enable the IPV6_RECVDSTOPTS socket option in order to receive the home address destination option:

```
int  on = 1;
setsockopt(fd, IPPROTO_IPV6, IPV6_RECVDSTOPTS, &on, sizeof(on));
```

Each Destination option header is returned as one ancillary data object described by a cmsghdr structure with cmsg_level set to IPPROTO_IPV6 and cmsg_type set to IPV6_DSTOPTS.

These options are then processed by calling the inet6_opt_next(), inet6_opt_find(), and inet6_opt_get_value() functions as defined in Advanced API for IPv6 sockets [1].

This document assumes that Mobile IPv6 applications will not be allowed to send Home Address Destination Option from the application level, as Mobile IPv6 kernel takes care of sending home-address option and routing header type 2.

The Destination options are normally constructed using the inet6_opt_init(), inet6_opt_append(), inet6_opt_finish(), and inet6_opt_set_val() functions, described in Section 10 of IPv6 Advanced API sockets [1].

4. Mobility Protocol Headers

Mobile IPv6 [2] defines a new IPv6 protocol header to carry mobility messages between Mobile Nodes, Home Agents and Correspondent Nodes. These protocol headers carry Mobile IPv6 Binding messages as well as Return Routability [2] messages. Currently the specification [2] does not allow transport packets (piggybacking) along with the mobility messages. Thus the mobility protocol header can be accessed through a IPv6 RAW socket. A IPv6 RAW socket that is opened for protocol IPPROTO_MH should always be able to see all the MH (Mobility Header) packets. It is possible that future applications may implement part of Mobile IPv6 signal processing at the

application level. Having a RAW socket interface may also enable an application to execute the Return Routability protocol or other future authentication protocol involving mobility header at the user level.

4.1 Receiving and Sending Mobility Header Messages

This specification recommends IPv6 RAW sockets mechanism to send and receive Mobility Header (MH) packets. The behavior is similar to ICMPV6 processing, where kernel passes a copy of the mobility header packet to the receiving socket. Depending on the implementation kernel may process the mobility header as well in addition to passing the mobility header to the application.
In order to comply with the restriction in Advance API for IPv6 [1] sockets, applications should set IPV6_CHECKSUM socket option with IPPROTO_MH protocol RAW Sockets. However, a Mobile IPv6 implementation that supports Mobile IPv6 API, must implement mobility header API checksum calculation by default at the kernel for both incoming and outbound path. A Mobile IPv6 implementation must not return error on IPV6_CHECKSUM socket option setting, even if the socket option is a NO-OP function for that implementation because it verifies the checksum at the kernel level. Mobility Header checksum procedure is described in Mobile IPv6 Protocol [2] specification.
Again, it is recommended that the applications set the IPV6_CHECKSUM socket option along with the RAW sockets for IPPROTO_MH protocol, for application portability.

As an example, a program that wants to send or receive mobility header protocol(MH), could open a socket as following:

```
fd = socket(AF_INET6, SOCK_RAW, IPPROTO_MH);

int offset = 4;
setsockopt(fd, IPPROTO_IPV6, IPV6_CHECKSUM, &offset,
     sizeof(offset));
```

For example, if an implementation likes to handle HOTI/HOT and COTI/COT message processing, it can do so by using IPv6 RAW Sockets for IPPROTO_MH at the application layer.
The same application may also set IPV6_RECVDSTOPTS socket option for receiving home address option in a binding update [2] from the mobile node.

5. Protocols File

Many hosts provide the file /etc/protocols that contains the names of the various IP protocols and their protocol numbers. The protocol numbers are obtained through function getprotoXXX() functions.

The following addition should be made to the /etc/protocols file, in addition to what is defined in section 2.4 of Advanced Sockets API for IPv6 [1].

The protocol number for Mobility is pending IANA (http://www.iana.orgassignments/protocol-numbers) assignment.

```
       ipv6-mh           62      # Mobility Protocol Header
```

6. IPv4-Mapped IPv6 Addresses

The same rule applies as described in section 13 of IPv6 Advanced API for Sockets [1]. Thus processing of IPv4-mapped IPv6 addresses for the Mobile IPv6 specific socket options are out of scope of this document.

7. Security Considerations

The setting of Home Address Destination option and route header Type 2 IPV6_RTHDR socket option may not be allowed at the application level in order to prevent denial-of-service attacks or man in the middle attacks by hackers.
Sending and receiving of mobility header messages are possible by IPv6 RAW sockets. Thus it is assumed that this operation is only possible by priviledged users. However, this API does not prevent the existing security threat from a hacker by sending bogus mobility header or other IPv6 packets using home-address option and Type 2 routing extension header.

8. References

[1]   Stevens, W. R, Thomas, M., Nordmark, E., Jinmei, T., "Advanced Sockets API for IPv6", RFC 3542, May 2003 April 19, 2002.

[2]   Johnson, D., Perkins, C., Arkko, J., "Mobility Support in IPv6" draft-ietf-mobileip-ipv6-24.txt, June, 2003.

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

9. Acknowledgement

Thanks to Brian Haley for the thorough review of this draft and many helpful comments. Keiichi Shima, Alexandru Petrescu, Ryuji Wakikawa, Vijay Devarapalli, Jim Bound, Suvidh Mathur and other mobile-ip working group members provided valuable input during the API design discussion. Vladislav Yasevich recommended default checksum calculation for mobility header API at the kernel level.

10. Authors' Addresses

Samita Chakrabarti
Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054, USA
Email: samita.chakrabarti@Sun.com

Erik Nordmark
Sun Microsystems Laboratories
180, avenue de l'Europe
38334 SAINT ISMIER Cedex, France
Email: Erik.Nordmark@sun.com