An Analysis of Mobility Handling in LIN6

Masahiro Ishiyama
Communication Platform Laboratory,
Corporate R&D Center,
Toshiba Corporation.

Mitsunobu Kunishi  Fumio Teraoka
Faculty of Science and Technology,
Keio University.

Abstract
LIN6 is a new protocol that supports mobility for IPv6. It supports two distinct hand-off mechanisms to accommodate various security needs. In this paper, we evaluate a prototype implementation of LIN6 and analyze its performance during hand-off. Our results show that LIN6 can complete a hand-off in about 50 milliseconds. We feel that this level of performance is adequate even for voice applications if supplemented by suitable error correction mechanisms. From our experiments, we also find out that the current IPv6 specifications do not adequately support fast mobility.

Keywords
mobile computing, network architecture, network protocol, node mobility, IPv6

1. Introduction
Mobile computing devices are evolving very rapidly, and nowadays a great number of people access the Internet through various types of mobile terminals. The number of Internet access points is expected to increase and wireless access links such as IEEE 802.11 and Bluetooth are expected to become more widely deployed. This is paving the way for a real mobile computing environment in which such portable devices can access the Internet and communicate with each other while on the move. Nobody wants to be limited to a usage model where only the receiving terminals are mobile and the sources of information (e.g. web servers) are fixed. Therefore, demand for a protocol that enables bi-directional peer-to-peer communications is emerging. IPv6 is the protocol that enables such communication; however, IPv6 does not include a mobility support mechanism—it needs an additional protocol to support mobile nodes. To support mobility in IPv6, Mobile IPv6 [1][2] has been proposed and is being discussed in IETF, but its standardization has not been completed yet and there are still some problems such as security issues and the existence of a single point of failure in its communication path.

In this paper "mobility" is defined as the functionality that satisfies the following two capabilities: 1) communication with nodes regardless of their location, 2) continuation of communication even if a correspondent node changes its location. From the mobility viewpoint, the fundamental problem of conventional network architectures is in the duality of the network address. For example, in both IPv4 and IPv6, an IP address has two meanings: the node identifier and the interface locator. The IP address specifies not only the identity of the node but also the point of attachment to the Internet. The IP address of the node changes if the node moves to another subnet. Consequently, the identity of the node is no longer preserved. This problem inherent in conventional network architectures is caused by assigning an address to the network interface of a node, not to the node itself. The locator of the network interface is regarded as the identifier of the node. In other words, there is no notion of a node identifier in the current Internet architecture.

We previously proposed the Location Independent Network Architecture (LINA) [3] which employs separation of identifier and locator to support node mobility. There are some network protocols based on the separation of the node identifier and the interface locator such as Xerox Internet Datagram [4], VIP [5, 6], and the GSE proposal [7] for IPv6. However, an entire network architecture covering the network layer to the application layer that considers node identity has never been implemented. The primary goal of LINA is to build an encompassing network architecture that is applicable to the design of practical protocols. We applied LINA to IPv6, and designed the protocol called LIN6. LIN6 provides mobility to IPv6 without impact on the existing IPv6 infrastructure and maintains compatibility with traditional IPv6. We also showed that LIN6 has several advantages compared to Mobile IPv6 in terms of performance and system stability. Our earlier work contains an overview of LIN6 and a preliminary performance evaluation of our prototype implementation.

In this paper, we describe details of LINA and LIN6, and analyze LIN6 mobility handling by using our prototype implementation. This paper is organized as follows. Section 2 introduces LINA and describes its communication model. Section 3 presents an application of LINA to IPv6, LIN6, and its design. Bootstrap, send
and receive procedures of LIN6 are detailed in Section 4. Section 5 describes the current implementation status of LIN6. Section 6 shows analysis of mobility handling of the LIN6 protocol. The advantages of LIN6 against Mobile IPv6 are discussed in Section 7.

2. LINA: Location Independent Network Architecture

This section introduces a new network architecture called LINA. LINA is based on the idea of separating the identifier and the locator of a node. In the application layer, a target node can be specified by its identifier in addition to the conventional model in which the target node is specified by the locator. The network layer is divided into two sublayers: the identification sublayer and the delivery sublayer. When the application specifies a target node, the identification sublayer maps the identifier to the corresponding locator, and then the delivery sublayer "embeds" the identifier in the locator. This embedding of identifiers into locators is done based on the new addressing model called embedded addressing.

2.1. Basic Concept

As described above, in conventional network architectures including IPv4/IPv6, the network address of a node denotes its identity as well as its location. This is a critical problem for supporting mobility at the network layer because there is no location independent identity for a mobile node. To solve this problem, LINA introduces two entities in the network layer to support node mobility.

- Interface locator: uniquely identifies the node's current point of attachment to the network. It is assigned to the network interface of a node and is used to route a packet to the network interface.

- Node identifier: signifies the identity of the node. It is assigned to the node itself and does not change even if the point of attachment to the network changes and a new interface locator is assigned to the network interface of the node.

Fig. 1 depicts the basic communication model based on LINA. The network layer is divided into two sublayers: the identification sublayer and the delivery sublayer. The identification sublayer converts the node identifier and the interface locator mutually while the delivery sublayer delivers the packet destined to the interface locator. An application program can specify a target node by either a node identifier (Fig. 1-(a)) or an interface locator (Fig. 1-(b)).

In case (a), an application wants to communicate with a node specified by the node identifier. This means that an application wants to communicate with a particular node regardless of its location. The transport layer maintains the connection with the pair of node identifiers. The identification sublayer in the network layer converts the node identifier to the appropriate interface locator, and the delivery sublayer delivers the packet. In this case, mobility is supported because the node identifier never changes even when the node moves.

In case (b), an application wants to communicate with a node located at the point specified by the interface locator. This means that an application wants to communicate with a node at the particular location of the network regardless of its identity. The transport layer maintains the connection between the pair of interface locators. The identification sublayer is bypassed, and the delivery sublayer delivers the packet. In this case mobility is not supported between nodes and the transport connection will reset if one of the nodes moves.

Specifying an interface locator at the application layer has several advantages. This feature can be used in following cases:

- An application wants to directly specify an interface. For example, a target node has several interfaces, and the application wants to communicate through a specific interface of the target node.

- An application wants to communicate with a particular node without identity. For example, a node wants to communicate with the node that is present at a particular location regardless of its identity.

2.2. Embedded Addressing Model

On the basis of the layering model described above, the network layer header in a packet should be divided into two headers in general protocol layering. However, this is not an efficient method since adding a new protocol header results in a greater overhead compared to existing architectures that use the interface locator as the node identifier. To solve this problem, we "embed" the node identifier in the interface locator. We call this addressing model the embedded addressing model.

The interface locator that follows this addressing model is called the ID-embedded locator. Although an ID-embedded locator is information for the delivery sublayer, it also implies a node identifier that is information for the identification sublayer. Thus, we can integrate the identification sublayer header into the delivery sublayer header by using ID-embedded locators in the network layer header, and the overhead issue is avoided accordingly. Detailed properties of the ID-embedded locator are described in the following section. However, a node still needs to determine the interface locator from the node identifier. In our model, a node simply refers to an association of the two which is managed outside of the packet header. The detail of this mechanism is described in Section 2.5.
2.3. Embedment and Extraction

Fig. 2 shows entities and operations that are used in the embedded addressing model. LIN A assumes that a node has one or more node identifiers that are assigned by an authority. It also has one or more interface locators when a node is connected to the network. Such locators are called current locators.

**Embedment** is an operation that determines the ID-embedded locator from the current locator and the node identifier. An **ID-embedded Locator** is also assigned to the interface of the node. Thus, the node is assigned not only current locators but also ID-embedded locators that are determined by performing embedment. The ID-embedded locator satisfies the following conditions:

1. An ID-embedded locator uniquely determines a node identifier without referring to other information.
2. It is possible to distinguish between an ID-embedded locator and a current locator.
3. An ID-embedded locator is a valid interface locator. That is, the format and the functions of the ID-embedded locator are equivalent to those of the interface locator and an intermediate node can deliver a packet even if the destination is specified by an ID-embedded locator.

We also introduce the concept of extraction that is the inverse operation of embedment. Extraction is an operation that determines a node identifier from an ID-embedded locator.

2.4. Generalized ID

In our concept described in Fig. 1, an application can specify not only a node identifier but also an interface locator. In this case, it is required that both the application layer and the transport layer handle both node identifier and interface locator. Thus, from an engineering standpoint, it would be easier to process, the node identifier and the interface locator in the same format.

For that reason, we introduce the concept of dedicated embedment. We introduce the dedicated locator that is the dedicated interface locator for dedicated embedment. The dedicated locator is a predefined well-known fixed value. The dedicated locator does not determine any physical point of the network whereas the ID-embedded locator and the current locator uniquely determine a physical point of attachment to the network.

In dedicated embedment, the dedicated locator is used for the current locator. The dedicated locator is a fixed value; thus the result of dedicated embedment has a one-to-one correspondence to a given node identifier. That is, the result can be used as an identifier for the node. We call the result of dedicated embedment the generalized identifier. Since the generalized identifier conforms to the format of an interface locator, an application layer and a transport layer does not need to discriminate between them, and hence the above issue is resolved.

2.5. Mapping: Resolving Interface Locator from Node Identifier

When a node performs embedment, the node needs to associate of the node identifier with the current locator.
of the node. This association is called *mapping*.

We introduce a function called a *Mapping Agent* that maintains this mapping. Designated Mapping Agents that are the Mapping Agents that maintain the mapping of a particular node identifier are introduced. That is, “Designated Mapping Agents of the node A” means that those Mapping Agents maintain the mapping of node A.

A node registers its mapping periodically to its designated Mapping Agents. It also registers a new mapping when the node changes its location on the network.

When a node performs embedding to determine an ID-embedded locator of a target node, and the node first determines the designated Mapping Agents of the target node and queries one of the designated Mapping Agents to acquire the mapping of the target node. Then the node can determine the current locator of the target node, and the node can perform embedding.

### 2.6. LINA Communication Model

We describe the detailed process of sending and receiving a packet in LINA, referring to Fig. 3 that shows the communication model of LINA, which is based on an application of the embedded addressing model to the basic communication concept shown in Fig. 1.

Upon sending a packet, an application specifies the destination with a generalized identifier for the target node. An identification sublayer examines whether the given destination is a generalized identifier or an interface locator. If the destination is a generalized identifier, the identification sublayer first performs extraction to obtain the node identifier, following which it determines designated Mapping Agents of the node identifier and queries the mapping of the node. Since it can derive the current locator of the target node when it obtains the mapping, it performs embedding and derives the ID-embedded locator of the target node. Then it passes the packet to the delivery sublayer with the ID-embedded locator. The delivery sublayer transmits the packet that is destined for the ID-embedded locator. If the destination is not a generalized identifier, the identification sublayer is bypassed.

When a packet is received, the delivery sublayer receives the packet and examines whether the specified source locator in the packet is an ID-embedded locator or not. If the source is an ID-embedded locator, the delivery sublayer passes the packet to the identification sublayer. The identification sublayer performs extraction and obtains the node identifier of the source node, following which it performs dedicated embedding to derive the generalized identifier of the source node. The identification sublayer informs the upper layer that the source locator of the packet is the generalized identifier. If the source is not an ID-embedded locator, the identification sublayer is bypassed.

### 3. LIN6: An Application of LINA to IPv6

We present LIN6, a protocol that supports mobility, by applying LINA to IPv6. For practical purposes, LIN6 is carefully designed to maintain compatibility with conventional IPv6 so that there is minimal impact on the existing IPv6 infrastructure. We show a method for applying the concept of embedding of LINA to IPv6, following which we describe a mechanism to determine a designated Mapping Agent.

#### 3.1. Embedded Addressing in LIN6

We first apply the concept of the ID-embedded locator of LINA to IPv6, where the locator is called a *LIN6 address*. Currently, addresses are assigned according to the Aggregateable Global Unicast Address (AGUA) [8] format, whose structure is shown in Fig. 4. In AGUA, the upper 64 bits of 128-bit IPv6 address indicate the network prefix to which the address belongs, and the lower 64 bits represent an interface ID. The interface ID is not required to be unique in the Internet but only in the subnet. Also, it is required to conform to the IEEE EUI-64 format [9].

![AGUA Format](image)

Figure 4: The format of Aggregatable Global Unicast Address and LIN6 address: In AGUA, the upper 64 bits represent the location of a subnetwork and the lower 64 bits represent the identifier of an interface, not of the node. In LIN6 address, although the upper 64 bits are the same as for AGUA, the lower 64 bits represent the node identifier, LIN6 ID.

Since this means that practical IPv6 subnetworks have a 64-bit network prefix length, the basic strategy in applying embedding to IPv6 is to use the lower 64 bits of the IPv6 address for the node identifier. That is, in LIN6, the address space of the node identifier is 64 bits. We call this 64-bit node identifier the *LIN6 ID*. Although a 64-bit address space is far smaller than the IPv6 128-bit address space, 64 bits can accommodate approximately $10^{16}$ nodes, which is considered sufficiently large to support LIN6 nodes.

This strategy satisfies condition (1) of the ID-embedded locator that is described in Section 2.3. To satisfy conditions (2) and (3), we form a LIN6 ID so that it could be identified as a LIN6 address, that is, we use part of the LIN6 ID to examine whether it is a LIN6 address or not. In this method, a LIN6 address can coexist with AGUA, that is, we can use the same prefix as in AGUA on a foreign network for the upper 64 bits of a LIN6 address,
and use a specially formed LIN6 ID for the lower 64 bits. The following method is an example of how this can be realized. In AGUA, the lower 64 bits are required to be constructed in EUI-64 format. In EUI-64, the upper 24 bits denote the Organizationaly Unique Identifier (OUI) assigned by IEEE, and the lower 40 bits are the value that is assigned by an administrator who is assigned an OUI. If an OUI is assigned for LIN6, a LIN6 address can be identified by examining the OUI part of the lower 64 bits of a given IPv6 address. This satisfies both condition (2) and condition (3) because a LIN6 address is a valid AGUA since a LIN6 ID completely follows the EUI-64 format in this method. Although the real address space of LIN6 ID decreases to 40 bits, this method does not have impact on existing IPv6 networks.

Since a LIN6 ID includes the specific OUI assigned for LIN6, strictly speaking, the LIN6 ID is in reality just the lower 40 bits, excluding the 24 bits assigned to the OUI. However, we simply call the entire 64-bit field including the OUI, the LIN6 ID, for convenience in following discussions.

### 3.2. Embedment in LIN6

In LIN6, the current locator of a LIN6 node is the IPv6 address assigned to the interface of the node. This address is generally an AGUA, and the mapping is an association between the LIN6 ID of the node and an AGUA assigned to the node at that time.

The operation of embedment in LIN6 is as follows. The upper 64 bits of a given IPv6 address which are the current interface address of a target LIN6 node are concatenated with the LIN6 ID of the target LIN6 node. The extraction operation simply draws out the lower 64 bits from a given LIN6 address. These operations are shown in Fig. 5.

### 3.3. Generalized Identifier in LIN6

We assume that a 64-bit-length network prefix for dedicated embedment which is called the LIN6 prefix has been allocated. The LIN6 prefix is a predefined, fixed value, and is expected that it is well known to all LIN6 nodes in advance. This is the same assumption that all IPv6 nodes know link-local prefix, local-loopback address, etc. It does not identify any physically existing subnetwork and is only used for dedicated embedment in LIN6, and thus we do not need routing information for it. The operation of dedicated embedment in LIN6 is simply to concatenate the LIN6 prefix to the LIN6 ID. The result of dedicated embedment is called LIN6 generalized ID, which corresponds to a generalized identifier in LINA. The LIN6 generalized ID completely conforms to the IPv6 address format, and existing IPv6 applications can specify this ID without any modifications as the destination of a correspondent node. Table 1 summarizes the correspondence between LINA and LIN6.
3.4. Finding Designated Mapping Agents

LIN6 needs mapping information to perform embedment. The address space of the LIN6 ID is 64 bits, and it might be very difficult to maintain mappings in a centralized single database.

In the Internet, the Domain Name System (DNS) resolves a similar issue. DNS is a distributed database that maps a Fully Qualified Domain Name (FQDN) to an IP address and vice versa, and provides other types of information related to an IP address and a FQDN. For example, consider the case in which one wants to know a FQDN associated with a particular IPv6 address. In this case, one sends a query with an IPv6 address as a key to any DNS server, and obtains the FQDN associated with the requested IPv6 address. This feature of DNS, called reverse lookup, works fine in the Internet.

DNS can be used to determine the mapping associated with a particular node identifier, that is, DNS servers can be used as Mapping Agents. However, using DNS servers as Mapping Agents presents a serious problem, since DNS is designed for handling "static" data. DNS assumes that the association between an IPv6 address and FQDN does not change frequently, hence DNS servers can cache data for load balancing. Since DNS is a very large scale distributed database, its success is mainly due to this caching mechanism. However, the content managed by Mapping Agents may change frequently, and thus DNS is inappropriate for Mapping Agents.

Consequently, we do not use DNS for the Mapping Agent, but introduce a new dedicated server. We call the node on which the server runs a Mapping Agent (MA). A network administrator of a LIN6 node decides the designated MA of the node, and then the administrator registers this relationship to DNS. That is, an association between the LIN6 generalized ID of the node and each IPv6 address of designated MAs of the node is registered in appropriate DNS servers. Consequently, a node can acquire the IPv6 addresses of designated MAs for the LIN6 node by querying DNS, specifying the LIN6 node’s LIN6 generalized ID as the key, as in the case of reverse lookup. The DNS server program must be modified so as to be able to handle information on MAs. However, this modification is only required for the master and the slave servers managing the LIN6 generalized ID. We do not need any modifications to root and intermediate DNS servers, since intermediate servers are only interested in a query name, and the query name is the LIN6 generalized ID which is compatible with an address in IPv6 address format. An association between a LIN6 node and its designated MAs will not change frequently, and consequently, this information is suitable for handling in DNS.

To acquire the mapping of a target LIN6 node, a node first sends a query to DNS to obtain information on designated MAs, and then sends a query to any of these MAs to acquire the mapping. An obtained mapping is cached in the node, and the caching area of mapping is called the mapping table. A mapping has a lifetime and is discarded when the lifetime expires. Also, addresses of MAs may be cached.

When a LIN6 node moves, it sends its mapping to one of the designated MAs of the node, and the designated MAs maintain consistency of the mapping by notifying each other of the received mapping. Since a LIN6 node can obtain the addresses of its designated MAs by querying DNS, LIN6 nodes do not need to know its designated MAs in advance. The communication mechanism of LIN6 is summarized in Fig. 6.

3.5. Hand-off of a Mobile Node

In LIN6, the current locator of a node is stored in the mapping entry as a cache on each correspondent node. Thus, if the node moves, a correspondent node needs to update the mapping to get hold of the new location of the node.

Mapping can be updated in one of two ways: 1) notification from the node that has moved, and 2) autonomous update by the correspondent node that is communicating with the node that has moved.

When a LIN6 node moves to a new location, it sends the new mapping to one of its designated MAs. In addition, the mobile node may send the new mapping to all correspondent nodes. This operation is called the Mapping Update. The mobile node can find its correspondent nodes by inspecting its mapping table.

If a mobile node sends the Mapping Update, some security mechanism such as IPsec [10] is needed to protect against spoofing attack. However, the mobile node might

---

Table 1: Correspondence between LINA and LIN6.

<table>
<thead>
<tr>
<th>LINA</th>
<th>LIN6</th>
</tr>
</thead>
<tbody>
<tr>
<td>node identifier</td>
<td>LIN6 ID, 64 bits but includes a specific OUI.</td>
</tr>
<tr>
<td>current locator</td>
<td>An IPv6 address that is generally an AGUA assigned to one of the interfaces of a LIN6 node.</td>
</tr>
<tr>
<td>ID-embedded locator</td>
<td>LIN6 address, completely follows AGUA format.</td>
</tr>
<tr>
<td>dedicated locator</td>
<td>LIN6 prefix, a 64-bit-length predefined prefix assumed to be assigned.</td>
</tr>
<tr>
<td>generalized identifier</td>
<td>LIN6 generalized ID, a concatenation of the LIN6 prefix and LIN6 ID.</td>
</tr>
</tbody>
</table>
not be able to use IPsec with the correspondent node. In
this case, the mobile node can send the Mapping Refresh
Request instead of the Mapping Update to the correspon-
dent node. The Mapping Refresh Request message does
not include the mapping. It only requests the correspon-
dent node to update its current mapping from the Map-
ping Agent. Thus, this message is free of any spoofing
attack. When a node receives a Mapping Refresh Re-
quest, it queries the mapping to the Mapping Agent and
obtains the new mapping.

However, correspondent nodes might not receive the
Mapping Update for some reason, for example, the Map-
ing Update packet is lost in the intermediate network,
or a mapping of the correspondent node is deleted from
the mapping table. LIN6 deals with this problem by us-
ing the ICMP Destination Unreachable Message (Dst Un-
reach) [11]. A Dst Unreach message is sent to the packet
sender from an intermediate router if reachability to the
destination is lost. If a LIN6 node receives a Dst Unreach
message when communicating with another LIN6 node,
it queries the designated MAs about the new mapping.
If the obtained mapping contains a new current interface
address, it can derive the new appropriate LIN6 address
of the correspondent LIN6 node and can continue to com-
municate. If a Dst Unreach message is received it can be
assumed that the correspondent LIN6 node is temporar-
ily disconnected from the network and is just in motion.
If the content of the obtained mapping is the same as
the cached one, or the mapping cannot be obtained, the Dst
Unreach message is notified to the upper layer. This situ-
ation signifies that the correspondent LIN6 node may be
off-line for a long time.

3.6. Compatibility with Traditional IPv6 Nodes

Let us consider the case when a LIN6 node wants to
communicate with a traditional IPv6 node. In this case,
an application specifies a traditional IPv6 address, not
a LIN6 generalized ID, as the destination, a situation
that corresponds to communicating via an interface loca-
tor in LIN6. The identification sublayer is consequently
bypassed, that is, no LIN6 specific operation such as
embedment is performed. As a result, this situation is
completely equivalent to traditional IPv6 communication.
Thus LIN6 nodes are able to communicate with tradi-
tional IPv6 nodes. However, when a LIN6 node com-
municates with traditional IPv6 nodes, mobility cannot
be supported since interface locators are used instead of
node identifiers.

4. Communication Example of LIN6

In this section, we describe LIN6 through an example
scenario depicted in Fig. 7. MN1 and MN2 are LIN6
nodes. G1, G2 and G3 represent network prefixes adver-
tised by a router in each network. i1 and i2 represent the
LIN6 IDs of MN1 and MN2, and e1 and e2 represent In-
terface IDs of MN1 and MN2 for AGUA. We use “+” as
the concatenation operator, that is, “G1+e1” represents an
IPv6 address that is a result of concatenation of network
prefix G1 and interface ID e1. “O” represents a LIN6
prefix, that is, “O+i1” represents a LIN6 generalized ID
of MN1. MA11 and MA12 are the designated Mapping
Agents of MN1, and MA21 is the designated Mapping
Agent of MN2. It is assumed that both MN1 and MN2
know their LIN6 identifiers and that they can obtain the
addresses of a DNS server.
4.1. Bootstrap

Let us consider that MN1 boots up and connects to network N1. MN1 registers the current mapping with one of its designated Mapping Agents as follows (see Fig. 8).

1. MN1 receives the Router Advertisement messages from a router on N1. At this point, MN1 acquires at least two global addresses: one is G1+e1, that is, a traditional AGUA, and the other one is G1+i1, that is, a LIN6 address.

2. MN1 sends a query to a DNS server to obtain the addresses of its designated MAs.

3. The DNS server replies with the addresses of the designated MAs of MN1, that is, MA11 and MA12. Recursive query of DNS is abbreviated from the figure.

4. MN1 selects a designated MA at will. Assuming that it chooses MA11, MN1 sends a registration packet of the mapping to MA11.

5. MA11 sends an acknowledgement of the registration back to MN1.

6. MA11 obtains the designated MAs of MN1 in a similar way, and finds that there is one more designated MA of MN1, that is, MA12. Then, MA11 notifies MA12 of the mapping to keep consistency of mapping of MN1.

4.2. Communication between LIN6 Nodes

Let us consider that MN1 communicates with MN2 that is on network N2. We assume that MN2 has already registered its mapping and G2+i2 as its LIN6 address. When an application on MN1 wants to communicate with MN2, it specifies LIN6 generalized ID of MN2, which is O+i2, as the destination address. Communication between MN1 and MN2 takes place as follows (see Fig. 9):

1. To acquire the current mapping of MN2, MN1 asks a DNS server about the designated MAs of MN2. As a result, MN1 finds MA21 is the designated MA of MN2.

2. MN1 sends a query to MA21, and acquires the current mapping for MN2. The mapping contains G2+e2, which is MN2’s current interface address.

3. MN1 performs extraction and obtains i2, the LIN6 ID of MN2, and then it performs embedding and derives G2+i2, the LIN6 address for MN2.

4. As a result, the destination address of the packet transmitted by MN1 is G2+i2, and the application recognizes the destination as O+i2. Similarly, the source address of the packet is G1+i1, the LIN6 address of MN1.

5. MN2 receives this packet and examines the source address field. Since the source address is a LIN6 address, MN2 performs extraction and obtains i1, the LIN6 ID of MN1 that is the source node of the packet. Then MN2 performs dedicated embedding and obtains O+i1, the LIN6 generalized ID of MN1, and then informs the application that O+i1 is the source address of the packet.

6. When MN2 sends a packet to O+i1, similar procedures are followed where the destination address in the packet MN2 transmits is G1+i1 and the source is G2+i2.

Thus, between MN1 and MN2 in the intermediate network, communication is perceived to be between G1+i1 and G2+i2, whereas on the node MN1, it is perceived to be between O+i1 and O+i2.

4.3. Hand-off of a LIN6 Node

Let us now consider the case when MN1 moves to Network N3 while communicating with MN2. When using Mapping Refresh Request, an update of MN1’s Mapping is performed as follows (see Fig. 10):

1. MN1 sends a Router Solicitation (RS) message, and it receives a Router Advertisement (RA) message from the router on N3. At this point, MN1 acquires G3+e1 as the AGUA and G3+i1 as the LIN6 address.
2. MN1 registers a new mapping (i1, G3+i1) to MA11.
3. MN1 inspects its mapping table to find current corresponding nodes. Since MN1 has cached MN2’s mapping (i2, G2+i2), MN1 recognizes that MN2 is one of them. MN1 sends a Mapping Refresh Request message to MN2 to let MN2 update its cached copy of MN1’s mapping.
4. When MN2 receives the Mapping Refresh Request, MN2 sends a Mapping Request to MA1 to update the mapping (we assume that a relationship between MN1 and MA11 is already cached at this point), and receives a new mapping for MN1.

Figure 10: Hand-off sequence using Mapping Refresh Request: MN1 registers a new mapping to its designated Mapping Agent, then sends a Mapping Refresh Request to let MN2 update its cached copy of MN1’s mapping.

5. Implementation

5.1. Implementation Status
We implemented a prototype of LIN6 on NetBSD/i386 1.5 with the KAME IPv6 stack. Fig. 11 shows an overview of our LIN6 implementation. Operations such as embedment, dedicated embedment and extraction, which are packet header manipulations, are implemented in the kernel, along with the mapping table. On the other hand, functions such as registration and acquisition of mapping are implemented in user space. This program is called the Mapping Resolver. We created a new socket, called the LIN6 socket, that is used by the mapping resolver to communicate with the kernel’s mapping table management functions. The Motion detector is an application program to detect the movement of the node and its new location. The reason for separating the Mapping Resolver for the Motion Detector is that we can easily modify the Motion Detector in order to try several motion detection mechanism which might have different performance. We also implemented the Mapping Agent as an application program.

Figure 11: LIN6 implementation overview. The Mapping resolver is an application program for mapping registration/acquisition. Operations such as embedment/extraction are implemented in the kernel. A LIN6 socket is used by the Mapping Resolver to communicate with the kernel.

5.2. Motion Detection Mechanism with IPv6
In our current prototype system, we use information from Layer 3 (i.e. IPv6 layer) to detect the movement of a mobile node, and we consider any change of the default router in the routing table of the kernel to be a movement of the mobile node. The Motion Detector which is shown in Fig. 11 watches changes of the default router. When the default router changes, it reads the current interface to be used and the new network location of the node, then
notifies the Mapping Resolver of the movement through the LIN6 socket. The selection and update of the default router is performed in the kernel and is a native function of the KAME IPv6 stack. When the Mapping Resolver receives a movement notification from the Motion Detector, it generates a current locator, i.e. a LIN6 address, and assigns the address to the current interface of the mobile node, then performs a mapping registration.

The discovery and selection of a default router is done using the Neighbor Discovery mechanism [12], which is part of the basic specification of IPv6. All IPv6 routers basically advertise their existence by a Router Advertisement (RA) message. A RA message includes information such as the network prefix to be used in the link. Thus a mobile node can determine its LIN6 address immediately when it receives a RA message. Although Neighbor Discovery specification states that RA be periodically sent to a link to which the router is connected, a node may request routers on the same link to send RA messages in order to avoid waiting for the next period. This request message is called Router Solicitation (RS). Since nodes are allowed to send a RS message to the multicast address to which all IPv6 routers must listen, a node does not need to know a router's address in advance.

It is possible that several routers are available on a link. Even if a node is connected to such link, the default router in the routing table does not change. The KAME IPv6 stack maintains all routers on the link as a candidates for the default router but selects only one router among the candidates, and it does not change the default router until the router becomes unreachable. Consequently, a change of default router properly indicates movement of a node, and the Motion Detector can detect movement correctly.

However, there is a performance implication with this detection method. Let us consider the case when a node moves to another network. The node can send a RS message, and might receive a RA message, but it cannot be determined if a node has moved by this RA message. This RA message may indicate that a new router is available of the link, but not as a result of the node moving to another link. In this case, a node must detect that the default router is unreachable. This detection is done with the Neighbor Unreachability Detection mechanism which is included in the IPv6 basic specification, and it normally takes 30 seconds in the worst case to detect unreachability.

We avoid this problem by using Layer 2 information if the link device is able to determine current connectivity status. For example, some Ethernet chips can notify the kernel of link carrier status, thus we are able to recognize a change of link. If the link device detects a change in a connected link, we disable the default router in the routing table before a RS message is sent to the link. Consequently, we can determine the new default router when the mobile node receives a RA message, and detect movement of the node more efficiently.

6. Protocol Evaluation of Mobility Handling

In this section, we evaluate the mobility handling mechanism of our prototype LIN6 implementation. We built an experimental network and performed a hand-off of a mobile node on the network, and we observed packets on the network by using tcpdump [13].

6.1. Experimental Network

Fig. 12 shows the configuration of our experiment network.

![Diagram](image)

Figure 12: The configuration of our experiment network. MN performs a hand-off from N2 to N3 while receiving VoIP packets from CN.

R1, R2, and R3 represent IPv6 routers. MN is a LIN6 mobile node, and CN is the correspondent node of MN. MA is a designated Mapping Agent of MN and CN. CN sends Voice over IP (VoIP) packets to MN, and we let MN move from network N2 to network N3 while MN is receiving VoIP packets from CN. We set the intervals between each VoIP packet to 20 milliseconds. All node are connected by 10BASE-T.

We logged packets observed at both network N1 and network N3. H represents the host who logs packets on the point Pa and Pb. From these logs we can analyze the behavior and performance of hand-offs in LIN6.

6.2. Experiment Results and Considerations

Fig. 13 shows the results of the experiment of a hand-off using Mapping Update. The number printed after the line number on each line indicates the time in seconds when the packet was received by tcpdump. The number in parentheses is the time difference from previous line. In our experiments, the IPv6 address of MA is already cached in MN and CN, thus DNS query packets are not
In Fig. 13, we find that it takes a very long time from (4) to (5), about 1036 milliseconds. This is a majority of the time that elapsed during hand-off. This is the time required for the DAD procedure. The IPv6 specification states that an address whose uniqueness on a link is being verified is not allowed to be used until the procedure is finished. The current KAME IPv6 stack only sends one Neighbor Solicitation packet for DAD in default. After MN sends the Neighbor Solicitation packet, it waits 1000 milliseconds, a value that is specified in the IPv6 specification as the default, for replying to the solicitation. The Mapping Resolver must wait for this procedure to complete. The mobile node is not allocated a valid global IPv6 address and thus no packets can be sent to it until this procedure ends. This is the reason for the long elapsed time from (3) to (4). This problem is not unique to LIN6 but also applies to other protocols that need to assign a new IPv6 address by the method that conforms to the IPv6 specification such as Mobile IPv6.

The time elapsed between sending a RS message (1) and receiving a RA message (2), takes 306 milliseconds. The reason for this time is that the router intentionally delays sending the RA message in response to the RS message. The IPv6 specification states that when an IPv6 router sends a RA message, the router must delay the transmission for a random amount of time in order to prevent routers on the same link from sending RA messages at exactly the same time. The default maximum time of delay is 300 milliseconds, and the KAME IPv6 stacks conforms to this value. This is the reason for the elapsed time until the RA message from the router is received. This problem is also confronted by all protocols that use RA messages, e.g. Mobile IPv6 in the case of using address autoconfiguration.

However, it is acceptable to decrease this random delay if the network has few routers on the same link, and it is possible to design such network in a real environment. Practically, address conflicts will not occur, especially in LIN6 because LIN6 addresses use LIN6 IDs in the interface ID part of IPv6 address format and the LIN6 ID is required to be unique on the Internet. It is possible that DAD be disabled on networks to support high speed hand-offs on certain networks.

Next, we performed the same experiment as in Fig. 13 except for the following two points. 1) We disabled the random delay when R3 sends a RA message in response to a RS message. 2) We set DAD count to zero on MN. This means that DAD is disabled.

The result of this experiment is shown in Fig. 14. In this case, the elapsed time from sending the RA message (1) to receiving the first VoIP packet (6) is about...
46 milliseconds, and we can estimate that MN drops two VoIP packets. Although this evaluation does not include the time needed to disconnect from the previous link, we think the extent of this overhead is acceptable enough since the packets lost can easily be recovered by error correction mechanisms such as Forward Error Correction (FEC). We measured the average time of the hand-off to be about 46.4 milliseconds with 10 tries.

We also tested the case of using Mapping Refresh Request. The result is shown in Fig. 15, with Random Delay and DAD are disabled.

(1) and (2) are RS and RA messages, and (3) and (4) are Mapping Registration and its ack, respectively. (5) is the Mapping Refresh Request to CN from MN. The Mapping Refresh Request is sent after receiving the ACK of Registration from MA (4), whereas MN does not wait for the ack when the Mapping Update is sent. The reason that MA waits for the ACK is to prevent CN from querying before MA learns the new mapping of MN and to ensure that CN can receive the new mapping of MN from MA. In line (5) we see a Mapping Refresh Request being transmitted. Notice it does not use an Authentication Header as it does not require IPsec as mentioned in Chapter 3.5. When CN receives the Mapping Refresh Request from MN, it sends the Mapping Query message to MA (5), and MA replies to it (6). At this point, CN now has the new mapping of MN that includes the new location of MN. In this case, the overhead of the LIN6 motion handling procedure mainly depends on three RTTs: MN-MA, MN-CN and CN-MA. The result shows that the overhead is about 55 milliseconds and the average is 49.7 milliseconds with 10 tries.

Our results show that Mapping Refresh Requests have performance implications over Mapping Updates, as the RTT between CN and MA increases. However, since Mapping Refresh Requests do not require IPsec mechanisms it can be used when CNs do not support IPsec and prove to be useful.

7. Related work: Mobile IPv6

In this section, we give an overview of Mobile IPv6 and its communication procedure. Then we compare LIN6 and Mobile IPv6.

7.1. Overview of Mobile IPv6

Mobile IPv6 provides transparent mobility at the IP layer [16], by using a fixed IPv6 address even if a mobile node changes its physical point of connection on the Internet.

In Mobile IPv6, a mobile node uses at least two IPv6 addresses, called home address and care-of-address, simultaneously. The home address is a fixed address, regardless of where the mobile node connects to the Internet, and care-of-address (CoA) is assigned at a newly visited link and thus indicates the mobile node’s current location in the Internet. When a correspondent node communicates with a mobile node that uses Mobile IPv6, the correspondent node identifies the mobile

Figure 14: Experiment 2: observing hand-off with DAD count 0, disabled Random Delay of RA, using the Mapping Update.

Figure 15: Experiment 3: observing hand-off with DAD count 0, disabled Random Delay of RA, using the Mapping Refresh.
node by home address. Since the home address does not change, whereas the CoA changes, when the mobile node moves, the correspondent node can continue communicating with the mobile node even if the mobile node changes its current point of attachment to the Internet.

As the home address is a normal IPv6 address, this address can be used as the destination IPv6 address of packets for the mobile node. Packets whose destination address is the home address are routed to the link to which the mobile node’s home address belongs. That link is called the home link. Mobile IPv6 introduces the Home Agent (HA). The HA is a router on a home link and maintains current location information of the mobile nodes. When the mobile node is away from the home link, the HA forwards the packets to the current CoA of the mobile node, and the mobile node receives packets destined for its home address which does not indicate the current location of the mobile node.

The mobile node periodically sends its current location information to the HA. This procedure is known as “registration” and the location information is called “binding”. The mobile node may notify correspondent nodes directly of its binding. A correspondent node that has a binding of the mobile node can send packets directly to the mobile node using the Routing Header without going through the HA.

### 7.2. Overview of the Communication Procedure of Mobile IPv6

Fig. 16 shows an overview of the communication procedure of Mobile IPv6. MN is a mobile node that uses Mobile IPv6 and communicates with CN, its correspondent node. HA represents the home agent of the mobile node, which is on the MN’s home link.

When MN changes its current point of attachment to the Internet, MN acquires its new CoA on the link, for example through stateless address autoconfiguration. Then MN registers its binding to HA. This operation is performed using a new IPv6 Destination option, called Binding Update Option, introduced by the Mobile IPv6 specification.

When MN sends a packet to CN, MN uses its CoA as the source address of the packet and uses a Home Address Option that includes the home address of the mobile node. When a packet includes the Home Address Option, its recipient must recognize that the packet is from the home address that is described in the Home Address Option, not from the source address of the packet. The Home Address Option is defined as one of IPv6 Destination Options and was introduced by Mobile IPv6.

MN may use the Binding Update Option in the packet to CN to notify the binding and CN may cache the binding when it receives a Binding Update Option. This cache is called the Binding Cache.

When MN sends a packet to CN, MN uses the home address of MN as the destination address. If CN has a Binding Cache for MN, CN may use a Routing Header [17] to send the packet directly to CoA instead of being routed to the home address of MN. A Routing Header, defined in the IPv6 base specification, allows the sender to specify an intermediate destination that the packet traverses before it is routed to its final destination.

If CN does not have a Binding Cache for MN, a Routing Header cannot be used, the packet is routed to the home link of the mobile node. The HA intercepts it, and forwards it to the CoA of the mobile node by using IPv6-in-IPv6 tunneling. In this case, the packet must go through an extra route: from CN to HA, and from HA to CoA, the current location of MN.

### 7.3. Comparison of LIN6 and Mobile IPv6

#### 7.3.1. Single Point of Failure: Mapping Agent and Home Agent

A LIN6 node needs a Mapping Agent (MA) to acquire mapping. Consequently, the MA is the single point of failure in LIN6. On the other hand, a home agent (HA) is the single point of failure in Mobile IPv6.

In order to enhance fault tolerance, managing of such agents in a distributed manner is a prerequisite. However, for Mobile IPv6, the location of the HA is dependent on the home address of a mobile node, that is, the HA must be placed on the subnetwork of the home address. It is possible to place multiple HAs in the same subnetwork, but placing them on a more distributed scale is very difficult. Thus, enhancing fault tolerance is difficult in Mobile IPv6.

On the other hand, the location of MAs is completely independent of the node identifier, hence designated MAs can be placed at any point in the network. Thus, LIN6 is more fault tolerant than Mobile IPv6.

#### 7.3.2. Overhead of Packet Header Length

Mobile IPv6 needs at least one extra destination option, a Home Address Destination Option, to inform the recipient of that packet of the mobile node’s home address. In addition, to route a packet to the mobile node through the optimal route, which means the packet is not routed through the HA, Mobile IPv6 uses the Routing Header, which is another IPv6 extension header. If two Mobile IPv6 nodes communicate with each other through an optimal route, both the Routing Header and the home address Option must be added to all of the packets that are exchanged between the nodes. The header length of the...
home address Option is at least 20 bytes and the Routing Header overhead for every packet is incurred when Mobile IPv6 nodes communicate with each other.

In recent years, Voice over IP (VoIP), a new Internet application, has been attracting a great deal of attention. Since VoIP requires exceedingly low delay, it uses small packets and much work [18] has been done on compressing not only the payload of packet but also headers to reduce delay. The overhead of packet header length that Mobile IPv6 bears could be a serious problem in such applications.

In contrast to Mobile IPv6, LIN6 does not incur any overhead on protocol header length. A LIN6 address contains not only a node identifier but also an interface locator. As a result, LIN6 supports the concept of separation of identifier and locator, but needs no extra protocol headers or options.

7.3.3. End-to-End Communication

In Mobile IPv6, if a node does not have the binding of a target correspondent mobile node, the packet destined for the mobile node is routed to the home address and the HA intercepts this packet and forwards it to the mobile node through IPv6-in-IPv6 tunneling. Thus, Mobile IPv6 does not always guarantee end-to-end communication.

On the other hand, in LIN6, mapping between the LIN6 generalized ID and the LIN6 address is performed only at the end nodes, once the mapping is obtained from the Mapping Agent. Thus, LIN6 does not require additional processing by intermediate nodes for packet delivery and always guarantees end-to-end communication without using tunnels.

When packets are not transferred end-to-end, issues such as decline in reliability arise. In Mobile IPv6, if the optimal route is not used, three paths of communication are required for it to function correctly. That is, a path from the mobile node to the correspondent node, from the correspondent node to the HA, and from the HA to the mobile node. Whereas, when end-to-end communication is guaranteed, a single path between the mobile node to the correspondent node is all that is required.

8. Conclusion

We analyzed mobility handling of LIN6 using our prototype implementation in our experimental network. LIN6 provides two distinct hand-off mechanisms, Mapping Update and Mapping Refresh Request, to accommodate various security needs, and we evaluated both mechanisms. The results showed that the Mapping Update and Mapping Refresh Request can support hand-off on average at 46.4 milliseconds and 49.7 milliseconds respectively. This is acceptable to many applications such as Voice over IP if suitable error correction mechanisms are used. Our experiments also showed that parts of current IPv6 specifications such as DAD and the random delay in Router Advertisement do not adequately support fast mobility.

Topics for future work include performance comparisons of Mobile IPv6 and LIN6 and experiments in wide-area commodity networks with practical applications such as video conferencing.

9. Acknowledgement

We are grateful to Dr. Francesco Stajano of Toshiba and particularly Dr. Atsushi Shionozaki of Sony for their valuable comments.

10. References


