PAPER *Internet Technology*

LINA: A New Approach to Mobility Support in Wide Area Networks

Masahiro ISHIYAMA[†], **Mitsunobu KUNISHI**[††], **Keisuke UEHARA**[†††], **Hiroshi ESAKI**[††††], *and* **Fumio TERAOKA**[†††††], *Nonmembers*

SUMMARY This paper presents a new network architecture called LINA that provides node mobility. The basic concept of LINA is separation of the node identifier and the interface locator. Although there are several protocols based on such a concept, they do not encompass issues that arise when dealing with an entire network architecture. LINA is a holistic architecture covering the network layer to the application layer to support node mobility. Overhead incurred by separation of the node identifier and the interface locator is minimized in LINA by introducing the embedded addressing model. This paper also presents a new protocol called LIN6 that supports IPv6 mobility as an application of LINA to IPv6. LIN6 is fully compatible with IPv6. It has several advantages in comparison to Mobile IPv6, e.g. less protocol overhead. Our prototype implementation of LIN6 shows minimal overhead compared to a conventional IPv6 implementation.

key words: mobile computing, network architecture, network protocol, node mobility, IPv6

1. Introduction

Mobile computing is becoming more common with advances in the performance of PDAs and growing popularity of mobile access devices. A great number of people access the Internet with their mobile terminals nowadays. Many kinds of access devices such as IEEE802.11 and Bluetooth are expected to be more diversified, and points of connection to the Internet are expected to be increased. Demands for a protocol that supports node mobility are emerging. To support communication with mobile nodes, especially for the Internet Protocol Version 4 (IPv4) and Version 6 (IPv6), many protocols have been proposed[1], [2]. However, these protocols are designed as extensions to IPv4 and IPv6 and consider mobility from a routing viewpoint. Although this approach is very useful considering its compatibility with existing protocols, when mobile computing becomes evermore diversified and becomes more widespread, a fundamental solution to support mobility form the architectural viewpoint for the long term.

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.

This paper proposes a new network architecture called the *Location Independent Network Architecture* (LINA). This architecture 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[3], VIP[4], [5], and the GSE proposal[6] for IPv6. However, an entire network architecture covering the network layer to the application layer that considers node identity has never been built. The primary goal of LINA is to build an encompassing network architecture that is applicable to the design of practical protocols.

We apply LINA to IPv6, and design the protocol called *LIN6*. LIN6 provides mobility to IPv6 without impact on the current IPv6 infrastructure that already exists and maintains compatibility with traditional IPv6. We also show that LIN6 has several advantages compared to Mobile IPv6[2] in terms of performance and system stability and is free from GSE issues of separation of identifier and locator that are pointed out in [10].

This paper is organized as follows. Section 2 proposes our network architecture called LINA and describes its communication model. Section 3 presents an application of LINA to IPv6, LIN6, and its design. Send and receive procedures of LIN6 are detailed in Section 4. Section 5 describes the current implementation

[†]Communication Platform Laboratory, R&D Center, Toshiba Corporation.
[†††]Graduate School of Media and Governance, Keio University.
[††]Faculty of Science and Technology, Keio University.
[††††]Information Technology Center, University of Tokyo.
[†††††]Sony Computer Science Laboratories Inc.

status of LIN6, and the result of preliminary performance evaluation. The advantages of LINA and LIN6 are discussed in Section 6 through the consideration of the issues of separation of identifier and locator based on the analysis of GSE[10] as well as the comparison with Mobile IPv6.

2. LINA: Location Independent Network Architecture

In this section, we propose the new network architecture, LINA. In LINA, the network layer that is described in the OSI seven-layer model is divided into two sublayers, one for identification and another for delivery. LINA introduces a specialized addressing model called *embedded addressing* to make protocols effective. In this section we outline the communication model of LINA. Then we describe the concept of LINA in detail, including the embedded addressing model.

2.1 The Basic Concept of LINA

As we 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 to provide mobility in 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: recognizes the identity of the node. It is assigned to a 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 a node.

Figure 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 (Figure 1-(a)) or an interface locator (Figure 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 by application 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 only present in the 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 presents overhead in comparison with the traditional network architecture that uses 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 *Embedded Addressing Model.*

The interface locator that follows this addressing model is called the *ID-embedded locator.* Although an ID-embedded locator is the information for the delivery sublayer, it implies a node identifier that is the information for 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 locater is 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.

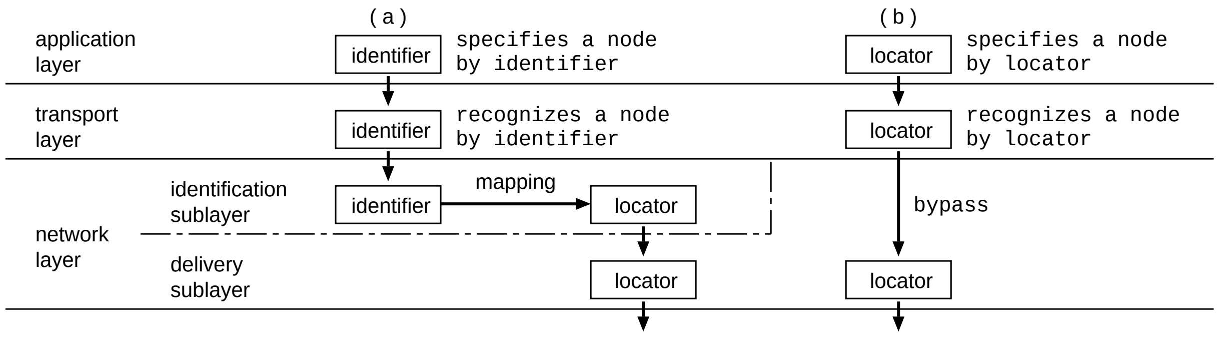

Fig. 1 Communication model: Network layer is divided into two sublayers to separate a locator and an identifier. An application can specify either a locator or an identifier to specify a correspondent node.

2.3 Embedment and Extraction

Figure 2 shows entities and operations that are used in the Embedded Addressing Model. LINA 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.*

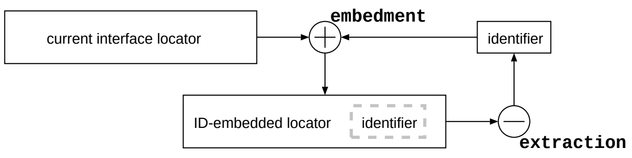

Fig. 2 Operations of the embedded addressing model: Embedment determines ID-embedded locator from the current locator and the node identifier, and extraction determines the node identifier from the ID-embedded locator

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 Figure 1, an application can specify not only a node identifier but also an interface locator. In this case, it is required for both the application layer and the transport layer to 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.

Dedicated embedment is a particular embedment operation that uses the dedicated locator 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, 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 the 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 embedment 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 embedment.

2.6 Communication Model in LINA

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

In this section, LINA is applied to IPv6, and a protocol called LIN6 to support mobility in IPv6 is designed. For practical purpose, 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 to apply the concept of embedment 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 ID-embedded locator of LINA to IPv6, where the locator is called a *LIN6 address.* The address assignment in present of IPv6 mainly follows the Aggregatable Global Unicast Address(AGUA)[7], whose structure is shown in Figure 4. In AGUA, the upper 64 bits of 128 bit IPv6 address indicates the network prefix to which the address belongs, and the lower 64 bits represents 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[13].

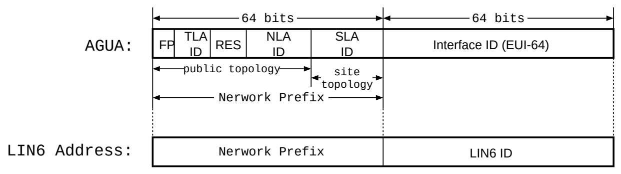

Fig. 4 The format of Aggregatable Global Unicast Address and LIN6 address: In AGUA, upper 64 bits represents the location of a subnetwork and lower 64 bits represent the identifier of an interface, not of the node. In LIN6 address, although upper 64 bits are same as AGUA, lower 64 bits represent the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

Since this means that practical IPv6 subnetworks have a 64 bit network prefix length, the basic strategy in applying embedment 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^{19} 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 visited network for the upper 64 bits of LIN6 address, and use a specially formed LIN6

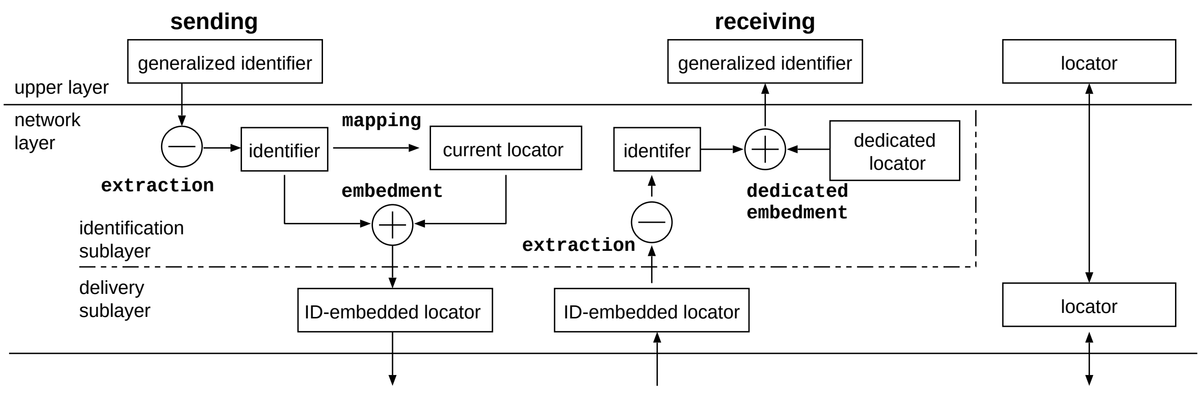

Fig. 3 Communication model of LINA: Upper layers identify a correspondent node with generalized identifier. The ID-embedded locator derived from the result of embedment is used for a packet delivery. Embedment is not performed when a locator is specified in upper layers.

ID for the lower 64 bits. The following method is an example of how this can be realized. In AGUA, lower 64 bits are required to be constructed in EUI-64 format. In EUI-64, the upper 24 bits denote the Organizationally Unique Identifier (OUI) that is assigned by IEEE, and the lower 40 bits are the value that is assigned by an administrator who received an assignment of OUI. If an OUI is assigned for LIN6 ID, we can identify a LIN6 address 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 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 existent IPv6 networks.

A LIN6 ID accordingly includes a specific OUI that is assigned for LIN6, thus the LIN6 ID is in reality just the lower 40 bits, excluding the 24 bits assigned to the OUI, in the strict sense. However, we simply call the entire 64-bit field including the OUI as 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 that is 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 operation of extraction is simply to draw out the lower 64 bits from a given LIN6 address. These operations are shown in Figure 5.

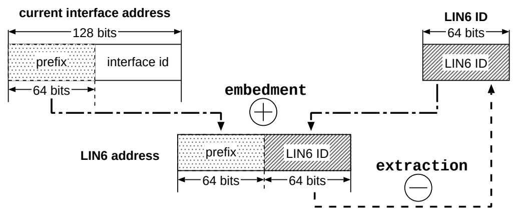

Fig. 5 Embedment in LIN6: A LIN6 address is derived by concatenating the prefix of the current IPv6 address and a node identifier.

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. LIN6 prefix is a predefined, fixed value, and is expected that all LIN6 nodes know it in advance. It does not identify any physically existent subnetwork and is only used for dedicated embedment in LIN6, 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* that corresponds to a generalized identifier in LINA. The LIN6 generalized ID completely conforms to the IPv6 address format, and existent 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)

Table 1 Correspondence between LINA and LIN6

LINA	LIN6
node identifier	LIN6 ID, 64 bits but includes a specific OUI.
current locator	An IPv6 address that is generally an AGUA assigned to one of the interfaces of a LIN6 node.
ID-embedded locator	LIN6 address, completely follows AGUA format.
dedicated locator	LIN6 prefix, a 64-bit length predefined prefix and is assumed to be assigned.
generalized identifier	LIN6 generalized ID, a concatenation of the LIN6 prefix and LIN6 ID.

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

We can use DNS to determine the mapping that is 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. It assumes that the association between an IPv6 address and FQDN does not change frequently, and so DNS servers thus can cache data for load balancing. Since DNS is a very large scale distributed database, its success is greatly due to this caching mechanism. However, the content managed by Mapping Agents may change frequently, 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) in LIN6. A network administrator of a LIN6 node decides the designated MA of the node, 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 to DNS specifying a LIN6 node's LIN6 generalized ID as the key, like as reverse lookup. The DNS server program must be modified 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 IPv6 address in format. An association between a LIN6 node and its designated MAs will not change frequently, and consequently, this information is suitable to handle in DNS.

To acquire a mapping of a target LIN6 node, a node first sends a query to DNS to obtain designated MAs, then sends a query to any one of the obtained 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 lifetime has expired. 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. We summarize communication mechanism of LIN6 in Figure 6.

3.5 Hand-off of a Mobile Node

When a LIN6 node moves, it sends the current mapping to one of its designated MAs. In addition, the node may send the current mapping to all correspondent nodes. This operation is called *Mapping Update*, which is similar to "Binding Update" in Mobile IPv6. The correspondent nodes can be obtained by inspecting its mapping table. However, correspondent nodes might not receive the mapping update for some reason, for example, the Mapping Update packet is dropped in the intermediate network, or a mapping of the correspondent node is deleted from the mapping table.

We deal with this issue by using ICMP Destination Unreachable Message[9](Dst Unreach). A Dst Unreach message is sent to a packet sender from an intermediate router if reachability is lost to the destination that is specified in the forwarding packet. If a LIN6 node receives a Dst Unreach message when communicating with another LIN6 node, it queries the mapping to one of the designated MAs of the correspondent LIN6 node to obtain a new mapping. If the obtained mapping contains a new current interface address, it can derive the new appropriate LIN6 address of the corespondent LIN6 node and can continue communication. If a Dst Unreach message is received it can be assumed that the correspondent LIN6 node is temporarily disconnected from network and is just in motion. If the content of the obtained mapping is the same as the cached one, or

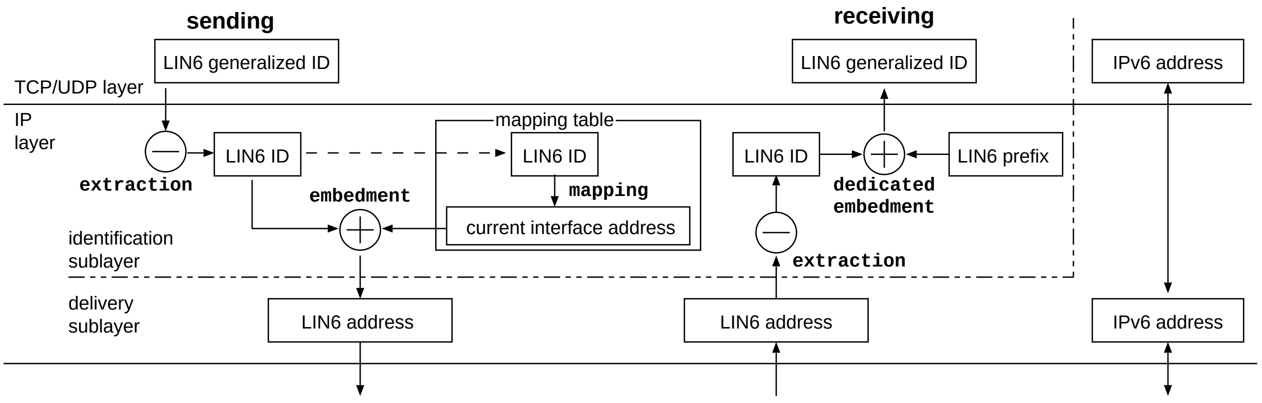

Fig. 6 Communication mechanism of LIN6: On sending, The LIN6 generalized ID is transferred to the LIN6 address by embedment with referring to the mapping table. On receiving, The LIN6 address is transferred to the LIN6 generalized ID by dedicated embedment. The LIN6 generalized ID is used in Upper layers while the LIN6 address is used to route a packet.

the mapping cannot be obtained, Dst Unreach message is notified to the upper layer. This situation 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 locator in LINA. 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 traditional IPv6 nodes. However, when a LIN6 node communicates with traditional IPv6 nodes, mobility cannot be supported since node identifiers are not used, but instead interface locators.

4. Communication Example of LIN6

In this section, we describe LIN6 through an example scenario depicted in Figure 7. MN1 and MN2 are LIN6 nodes. G1, G2 and G3 represent network prefixes advertised by a router in each network. i1 and i2 represent the LIN6 IDs of MN1 and MN2, e1 and e2 represent Interface IDs of MN1 and MN2 for AGUA. We use “+” as the operator of concatenation, that is, “G1+e1” represents an IPv6 address that is a result of concatenation of network prefix G1 and interface ID e1. “O” represents 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.

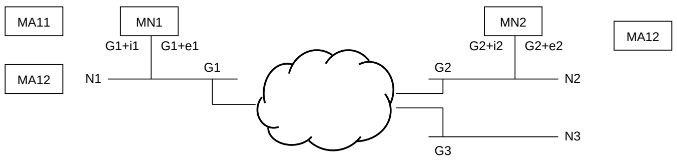

Fig. 7 Sample configuration

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 Figure 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 the addresses of the designated MAs of MN1, that are 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.

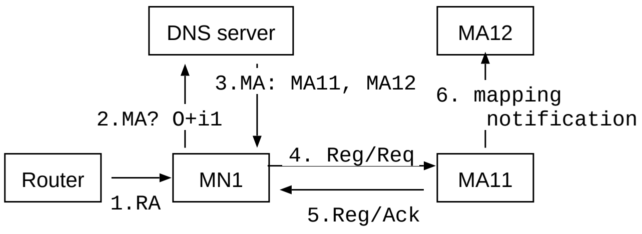

Fig. 8 Bootstrap sequence: MN1 queries a DNS server about its designated MAs and registers its current mapping to one of the designated MAs.

4.2 Communication between LIN6 Nodes

Let us consider that MN1 communicates with MN2 that is on network N2. We assume that MN2 already registered its mapping and it has the 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 Figure 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, then it performs embedment and derives G2+i2, the LIN6 address for MN2.
4. As a result, the destination address of the packet that is transmitted by MN1 is G2+i2, while 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 embedment and obtains O+i1, the LIN6 generalized ID of MN1, 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 packet that 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, while 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. MN1 acquires G3+e1 as the AGUA and G3+i1 as the LIN6 address. MN1 registers a new mapping (i1, G3+e1) to one of the designated MAs of MN1. Then, MN1 inspects its mapping table. Since MN1 has the mapping (i2, G2+e2), MN1 can send a mapping update to MN2. When MN2 receives this, MN2 updates the mapping for MN1. As the result of this, MN1 and MN2 can continue communicating.

5. Implementation Status and Evaluation

5.1 Implementation Status

We implemented a prototype system of LIN6 on NetBSD/i386 1.4.1 with the KAME IPv6 stack. Figure 10 shows the overview of our LIN6 implementation. Operation such as embedment, dedicated embedment and extraction, which are a sort of 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 is implemented in user space. This program is called the *Mapping Resolver*. We created a new socket, called the *LIN6 socket*, that is used by mapping resolver to communicate with the kernel's mapping table management functions. We also implemented the Mapping Agent as an application program. Obtaining information on designated MAs by interacting with DNS servers and mapping updates have not been implemented yet.

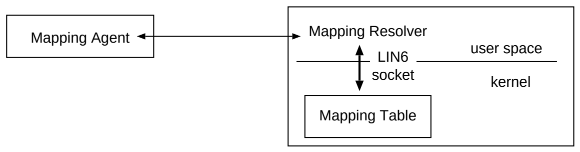

Fig. 10 LIN6 implementation overview: The Mapping resolver is an application program for mapping registration/acquisition. Operations such as embedment/extraction are implemented in the kernel. LIN6 socket is used by the Mapping Resolver to communicate with the kernel

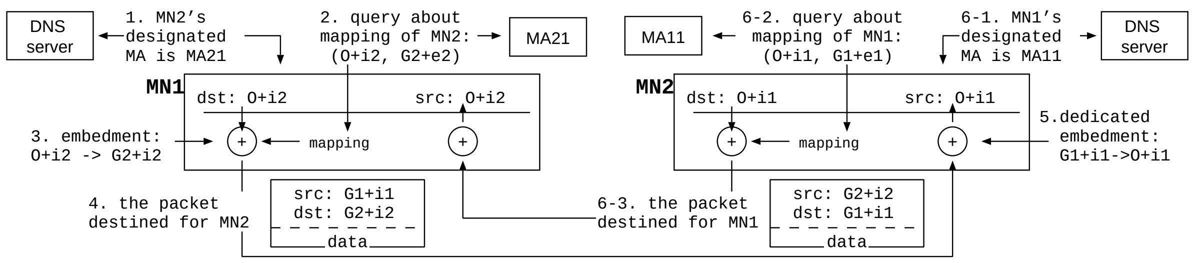
DNS server
1. MN2's designated MA is MA21
2. query about mapping of MN2: (O+i2, G2+e2)
MA21
MN1
dst: O+i2
src: O+i2
3. embedment: O+i2 -> G2+i2
mapping
4. the packet destined for MN2
src: G1+i1
dst: G2+i2
data
MA11
6-2. query about mapping of MN1: (O+i1, G1+e1)
6-1. MN1's designated MA is MA11
DNS server
MN2
dst: O+i1
src: O+i1
5.dedicated embedment: G1+i1->O+i1
mapping
6-3. the packet destined for MN1
src: G2+i2
dst: G1+i1
data

Fig. 9 Example of LIN6 communication: Upper layers of MN1 and MN2 identify each other with LIN6 IDs, and LIN6 addresses are used as source and destination address in packets that flow in the Internet.

5.2 Performance Evaluation

Evaluation was performed on PC/AT compatible machines with a Pentium-III 550MHz processor. The processing times described below were measured with the Pentium performance register.

Table 2 shows the processing time required to input and output packets using IPv6 and LIN6. For LIN6, we supplied 10 mapping entries including the target mapping in the kernel mapping table.

Table 2 Comparison of traditional IPv6 and LIN6 input/output processing time(microsecond).

	IPv6	LIN6
Input	0.64	0.65(+1%)
Output	2.77	4.05(+46%)

For output, in LIN6, a large overhead is incurred while the overhead is very small for input. The reason is that, in output, LIN6 needs to search for the mapping of the target packet to perform embedment while this is not required for input. Since IPv6 transmission on Ethernet takes place in the order of a millisecond, the LIN6 output overhead is negligible from the user's viewpoint.

When a node does not have the mapping of the target node, the kernel needs to ask the mapping resolver in user space to acquire the mapping through the LIN6 socket. We also measured the performance of a LIN6 socket, that is, the interaction between the kernel and the mapping resolver. The result of this measurement was 349.1 microseconds. Practically, we need to add round-trip time to a designated MA to this overhead. The round-trip time would be on the order of milliseconds, so this overhead of the interaction between the kernel and user space is also negligible.

We also measured the processing time of search a random mapping from the mapping table with 1000 entries. The result of this measurement was 0.30 microseconds.

6. Considerations

6.1 Separation of Node identifier and Interface Locator

There have been considerations and discussions on the separation of identifier and locator on network protocols. For IPv6, this separation is proposed in "GSE"[6]. GSE proposes to divide an IPv6 address to mainly two parts, the upper 64 bits for routing and the lower 64 bits for unique identification of an end point.

LIN6 undoubtedly differs from GSE in following two ways:

- In LIN6, a node identifier is assigned to the node itself while identifier is assigned to a network interface in GSE.
- In LIN6, the entire LIN6 address is used for the locator and it is not modified at intermediate routers while the upper 64 bits of IPv6 address that is used for routing is modified at intermediate routers in GSE.

There is also an analysis[10] of GSE. It points out many issues of GSE, for example, 1) there is no scalable method to resolve a locator from an identifier, 2) it is considered to be difficult to provide reliable TCP handshake, 3) it is difficult to assure that traffic from a source identified only by an identifier actually comes from the correct source.

LIN6 is not affected by these problems. LIN6 provides scalable method to resolve a LIN6 address from LIN6 generalized ID through the Domain Name System to address issue 1). [10] also points out it is infeasible to update the DNS record frequently, however, in LIN6 even if the node moves, there is no need to update the DNS record because DNS maintains a relationship between the node and the designated MAs, not the mapping of the node. 2) and 3) are caused by the GSE feature of rewriting the routing part of an IPv6 address in intermediate routers. In LIN6, the address in an outstanding packet is never rewritten by intermediate routers. Thus LIN6 is free from these issues, although secure communication must be supported between a LIN6 node and its designated MAs.

6.2 Comparison of LIN6 and Mobile IPv6

6.2.1 Single Point of Failure: Mapping Agent and Home Agent

A LIN6 node needs MAs to acquire mapping consequently, MA is the single point of failure in LIN6. On the other hand, Home Agent(HA) is the single point of failure in Mobile IPv6.

In order to enhance fault tolerance, managing such agents in a distributed manner is a prerequisite. However, for Mobile IPv6, the location of HA is dependent on the Home Address of a mobile node, that is, 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 in 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, so designated MAs can be placed at any point in the network. Thus LIN6 can provide higher fault tolerance in comparison with Mobile IPv6.

6.2.2 Overhead of Packet Header Length

Mobile IPv6 needs at least one extra destination option, the 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 an optimal route, which means the packet is not routed through a HA, Mobile IPv6 uses the Routing Header[11], which is another IPv6 extension header. Let us consider 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 is at least 24 bytes. A 44-bytes packet header overhead for every packet is incurred when Mobile IPv6 nodes communicate each other.

In recent years, Voice over IP(VoIP) has been attracting a great deal of attention as the new application for the Internet. VoIP requires exceedingly low delay, accordingly it uses small packets and there has been extensive work[12] presented to compress 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 application.

In contrast to Mobile IPv6, LIN6 does not have an overhead of protocol header length in comparison with traditional IPv6. A LIN6 address contains not only a node identifier but also an interface locator. As a result, while LIN6 supports the concept of separation of identifier and locator, LIN6 needs neither extra protocol headers nor options.

6.2.3 End-to-End Communication

In Mobile IPv6, if a node does not have a binding of a target correspondent mobile node, the packet that is destined for the mobile node is routed to the home address and a HA intercepts this packet and forwards it to the mobile node through IPv6-in-IPv6 tunnelling. Thus, Mobile IPv6 does not always guarantee End-to-End communication. On the other hand, a LIN6 node converts LIN6 generalized ID to LIN6 address individually, LIN6 always guarantees End-to-End communication.

When packets are not transferred end-to-end, issues such as decline in reliability arises. 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 corespondent 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.

7. Conclusion

This paper proposed a new network architecture called LINA. LINA employs the notion of separating the identifier and locator to support mobility and integrates the conceptually separated sublayers into a single packet header based on embedded addressing model. The embedded addressing model enables LINA to avoid from packet length overhead.

We designed LIN6, which is an application of LINA to IPv6, that provides node mobility to IPv6. LIN6 is compatible with traditional IPv6 and does not affect the existent IPv6 infrastructure. A prototype system of LIN6 was implemented on NetBSD/i386 with the KAME IPv6 stack, and the measured overhead incurred was very small and acceptable. We showed that LIN6 does not present the issues that arise in GSE, and that LIN6 has several advantages when compared with Mobile IPv6 in terms of a header length overhead and system stability.

Several topics that are candidates for future work include: the implementation of Mapping Update and finding designated MAs through DNS, a detailed mechanism for integration with IP security protocols, and consideration of fast hand-off.

References

[1] Perkins, C.: IP Mobility Support, RFC 2002 (1996).

[2] Johnson, D. B. and Perkins, C.: Mobility Support in IPv6, Internet-draft, IETF (1999) (Work in progress).
[3] Xerox.: Internet Transport Protocols. XSIS 028112, Xerox Corp. (1981).
[4] Teraoka, F., Uehara, K., Hideki, S. and Murai, J.: VIP: A Protocol Providing Host Mobility, Communications of the ACM, Vol. 37, No. 8, pp. 67-75 (1994).
[5] Teraoka, F.: Mobility Support with Authentic Firewall Traversal in IPv6, IEICE Transaction on Communication, vol.E80-B, no.8, (1997).
[6] O'Dell, M.: GSE - An Alternate Addressing Architecture for IPv6, Internet-draft, IETF (1997) (Work in progress).
[7] Hinden, R., O'Dell, M. Deering, S.: An IPv6 Aggregatable Global Unicast Address Format, RFC2374, IETF (1998).
[8] Hinden, R., Deering, S.: IP Version 6 Addressing Architecture, RFC2373, IETF (1998).
[9] Conta, A. and Deering, S.: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification, RFC2463, IETF (1998).
[10] Crawford, M. et al: Separating Identifiers and Locators in Addresses: An Analysis of the GSE Proposal for IPv6, Internet-draft, IETF (1999) (Work in progress).
[11] Deering, S., Hinden, R.: Internet Protocol, Version 6 (IPv6) Specification, RFC2460, IETF (1998).
[12] Douskalis, B: IP Telephony, Hewlett-Packard Professional Books, Reading (2000).
[13] Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority, `http://standards.ieee.org/regauth/oui/tutorials/EUI64.html`, IEEE (1997).