LIN6: A New Approach to Mobility Support in IPv6

Mitsunobu Kunishi
Graduate School of Science and Technology,
Keio University

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

Keisuke Uehara
Keio Research Institute at SFC,
Keio University

Hiroshi Esaki
Information Technology Center,
University of Tokyo

Fumio Teraoka
Sony Computer Science
Laboratories Inc.

Abstract - **This paper describes a new network architecture called *LINA* (Location Independent Network Architecture) that supports node mobility. It also describes a new protocol called *LIN6* that is based on LINA. LIN6 is fully compatible with IPv6 and free from the problems of Mobile IPv6. Our prototype implementation of LIN6 has a much lower processing overhead in comparison with conventional IPv6.**

Keywords: **Network Protocol, Mobility, IPv6**

I. Introduction

Although Mobile IPv6[1] is being standardized in IETF to support IPv6[2] mobility, it has several problems. First, Mobile IPv6 has large header overhead since it makes use of extension headers. For example, in the case of communication between two mobile nodes, the header overhead becomes 48 bytes, versus the 40 bytes of the IPv6 base header. Second, the location of the Home Agent (HA) is restricted by the home address of the mobile node. This reduces tolerance to failures of the network and/or of the HA itself. Third, Mobile IPv6 is potentially inconsistent with IP security protocols[3]. These problems basically result from the lack of consideration taken in designing IPv6's network architecture.

We describe a new network architecture called LINA that provides node mobility. LINA is based on separation of the node identifier and the interface locator. LINA avoids incurring header overhead. It also avoids vanishing backward compatibility caused by separation of the node identifier and the interface locator. We describe a new network protocol called LIN6 to support IPv6 mobility as an application of LINA to IPv6. LIN6 solves several of the problems of Mobile IPv6.

II. LINA: Location Independent Network Architecture

II.1 Basic concept

In conventional network architectures including IPv4/IPv6, the network address of a node denotes its location and also its identity. This feature causes a critical problem when it comes to providing mobility in the network layer because there is no location independent identity for a mobile node. To avoid this problem, LINA introduces two new concepts that are *node identifier* and *interface locator*. The node identifier recognizes the identity of the node. The node identifier does not depend on its attachment point to the network interface. The interface locator denotes the current point of attachment to the network. It is assigned to the network interface of a node and is used to route packet to the network interface. The node identifier is immutable whereas the interface locator changes when the node moves.

II.2 Node Identifier and Interface Locator

LINA uses a *Mapping Agent*(MA) to map the node identifier to the interface locator. The relation between the node identifier and the interface locator is called a *mapping*. A node registers its mapping periodically with its mapping agents. It also registers a new mapping when the node changes its location on the network.

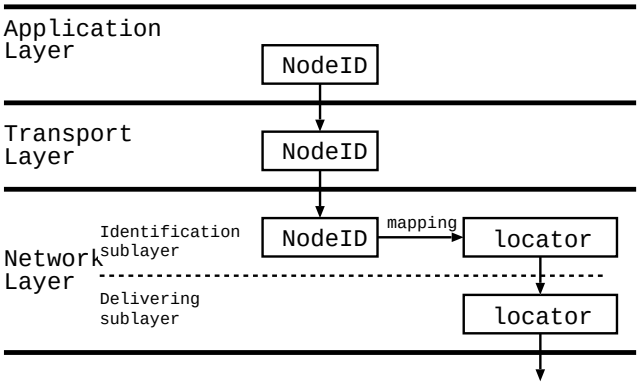

Figure 1: Network architecture model

II.3 Network Layer Model

The transport and upper layers can communicate with the mobile node by specifying the node identifier regardless of the interface location. This identifier is called *generalized identifier*. The conventional network layer is divided into two sub layers: the *identification sub layer* and the *delivering sub layer* (Figure 1). The generalized identifier, which is passed to network layer from the transport layer is mapped to the interface locator in the identification sub layer by querying the MA. The delivering sub layer delivers the packet in accordance with the interface locator. However, the use of two headers in each sub layer is inefficient and causes a header overhead like that of Mobile IPv6. LINA integrates the identification sub layer into the delivery sub layer header and thus avoids header overhead. The node identifier can be obtained from the locator.

III. LIN6: An Application of LINA to IPv6

III.1 Addressing Method

A applying LINA to IPv6, require a new network protocol, which we call LIN6. LIN6 is designed so as to maintain compatibility with conventional IPv6, i.e., so that there is minimal impact on the existing IPv6 infrastructure. The LIN6 addressing method is based on IPv6 Aggregatable Global Unicast Address(AGUA)[4], in which the upper 64 bits of the 128 bits IPv6 address indicates the network prefix to which the address belongs, and the lower 64 bits represents the interface ID. In LIN6, the interface locator is 128 bits long, and the node identifier is 64 bits long.

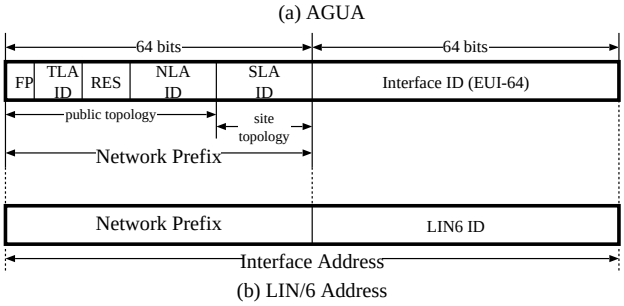

Figure 2: Addressing method

Figure 2 shows the relation between an AGUA and a LIN6 address. The LIN6 address indicates the current locator of the LIN6 node. This address is generally an AGUA. The upper 64 bits of the LIN6 address is the current network prefix which indicates the current interface locator and lower 64 bits of the LIN6 address is the node identifier(*LIN6 ID*). The LIN6 address is created by "embedding" the LIN6 ID in the lower part of the IPv6 AGUA structure. The LIN6 ID is obtained by from the lower 64 bits of the LIN6 address. These operations are called *embedment* and *extraction*.

LIN6 introduced *LIN6 generalized identifier* for the transport and upper layers identifier. This identifier is as same as the generalized identifier of LINA and immutable whereas the LIN6 address changes when the node moves.

III.2 Communication Mechanism

Figure 3 shows the LIN6 sending/receiving communication mechanism.

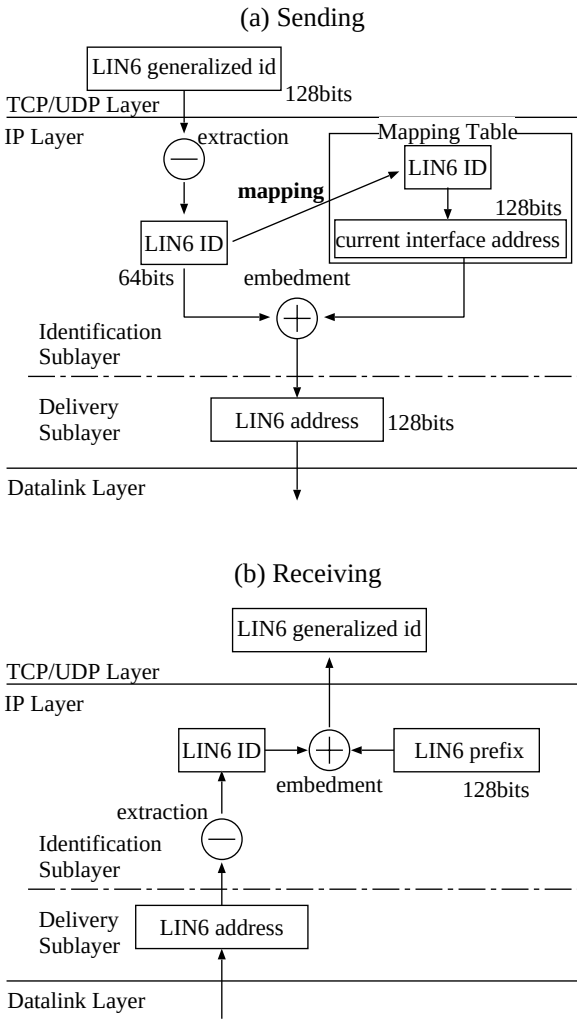

Figure 3: LIN6 communication mechanism

On sending(Figure 3(a)), the LIN6 generalized identifier is transferred to the LIN6 address by using the following procedure. The identification sub layer performs extraction from the LIN6 generalized

identifier to obtain the LIN6 ID and sends a query to MAs to obtain the mapping using the LIN6 ID as the key. Since it can derive the current locator(interface address) when it obtains the mapping, it performs the embedment procedure and derives the LIN6 address. LIN6 address is used for packet delivery in the delivery sub layer.

On receiving(Figure 3(b)), the delivery sub layer receives the packet and passes the packet to the identification sub layer. The identification sub layer performs the extraction procedure. The LIN6 ID performs the embedment procedure with a 64 bit fixed network prefix (*LIN6 prefix*) to obtain the LIN6 generalized identifier of the source node. Thus, the LIN6 address is transferred to the LIN6 generalized identifier.

III.3 Compatibility with Conventional IPv6

When a LIN6 node wants to communicate with a conventional IPv6 node, an application specifies a conventional IPv6 address, not a LIN6 generalized identifier, as the destination address. The identification sub layer does not execute LIN6 specific operations such as embedment. As a result, the LIN6 node communicates as the equivalent of a conventional IPv6 node. However, when a LIN6 node communicates with conventional IPv6 nodes, mobility cannot be supported since node identifiers are not used.

III.4 Distinguishing LIN6 addresses from conventional IPv6 addresses

LIN6 addresses can't distinguish from conventional IPv6 addresses. The LIN6 ID is used so that a LIN6 address can be identified as such, that is, part of the LIN6 ID to identify a LIN6 address. 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 the LIN6 address, and we can use a LIN6 ID for the lower 64 bits. The lower 64 bits of AGUA must be in EUI-64 format[2]. The upper 24 bits of EUI-64 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[5]. If an OUI is assigned for a LIN6 ID, we can identify a LIN6 address by examining the OUI part of the lower 64 bits of a received packet address.

III.5 Communication Example

Figure 4 depicts the LIN6 communication procedures. There are two mobile nodes, *MN1* and *MN2*, which are assumed to have registered their mappings with MA. The communication procedure is as follows:

1. MN1 sends a query to MA to obtain the current locator of MN2.

2. MA returns the current locator of MN2 to MN1.

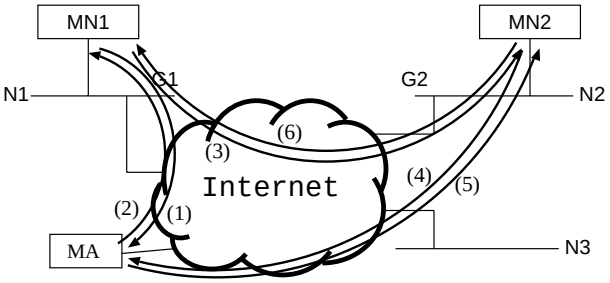

Figure 4: LIN6 communication procedure

3. MN1 creates the LIN6 address of MN2 and sends a packet.

4. MN2 also sends a query to MA to obtain the current locator of MN1.

5. MA returns the current locator of MN1 to MN2.

6. MN2 creates the LIN6 address of MN1 and sends a packet.

MN2 caches the current locator of MN1. When MN1 moves to another subnet and get a new locator, it sends a mapping update packet to MA and MN2.

III.6 Finding a Mapping Agent

We purpose using a scalable method for finding MAs. LIN6 makes use of the Domain Name System (DNS) to locate the MA(s) of a mobile node. DNS maintains the list of addresses of MAs of a mobile node. When a correspondent node wants to send a packet to a mobile node for the first time, the correspondent node sends a query to the DNS server and obtains the list of addresses of the MAs of the mobile node.

Figure 5 shows the bootstrap sequence for a LIN6 node registering its current locator with its MA.

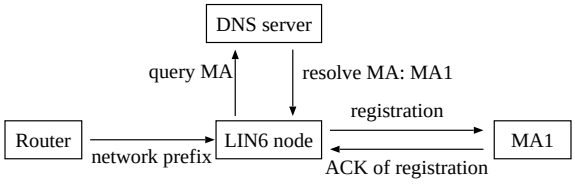

Figure 5: Bootstrap sequence

III.7 Mapping Update

When a LIN6 node moves from from one network to another, it gets a network prefix and sends the current mapping to its MAs. The LIN6 node also sends the current mapping to all correspondent nodes. This

operation is called *Mapping Update*. When a correspondent node receives a mapping update message and obtains by inspecting its mapping cache. mapping update messages should have an *Authentication Header*(AH)[6].

IV. Implementation

IV.1 Implementation Model

We implemented a prototype system of LIN6 on NetBSD/i386 with KAME IPv6 stack. Figure 6 shows an overview of our implementation.

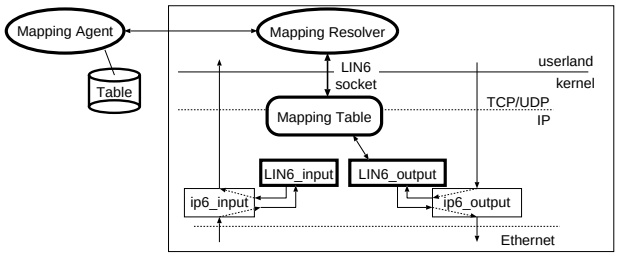

Figure 6: LIN6 implementation model

LIN6 basic operations such as embedment the node identifier to interface locator and extracting the node identifier from the interface locator were implemented in the kernel with the mapping cache table. LIN6 mapping functions such as registration and acquisition of mapping were implemented in application program. This program is called *Mapping Resolver*. Between the mapping resolver and kernel's mapping table management functions communicate through a socket called *LIN6 socket*. We implemented the Mapping Agent as an application program, and created a DNS pseudo entry type to relate node identifier to MAs.

IV.2 Operation

For input, the network layer calls `ip6_input()`. LIN6 extracts the LIN6 ID from the received packet. If the destination address is a LIN6 address, it calls `LIN6_input()` and gets the generalized identifier. This process is delivered to upper layer after getting generalized identifier. For output, the IP upper layer calls `ip6_output()`. If the destination address is a generalized identifier, it calls `LIN6_output()`. `LIN6_output()` extracts the LIN6 ID from the generalized identifier and obtains the current locator of the destination node. Mapping information is maintained in the mapping table in the kernel as a cache. If an entry is found in the mapping table, the LIN6 ID is embedded in the current locator. If an entry can't be found in the mapping table, the mapping resolver is called via LIN6 socket with the LIN6 ID. The mapping resolver sends a query to the MA and obtains a current locator for the LIN6 ID. The mapping resolver give the current locator to the kernel via LIN6 socket. If destination address is not a LIN6 address, it performs the conventional IPv6 operation since it doesn't call either `LIN6_input()` or `LIN6_output()`.

V. Evaluation

We measured packet processing overhead as a basic performance test of our LIN6 prototype implementation, The mapping table manipulations were addition, deletion, and search of a mapping while increasing the number of mapping entries in the mapping table. The evaluation was performed on PC/AT compatible machines with a Pentium-III 550MHz processor. The processing times were measured with the Pentium performance register.

V.1 Packet Processing Overhead

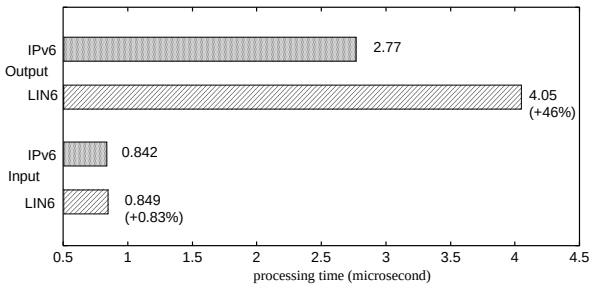

Figure 7: Operation overhead

Figure 7 shows the processing time of transmission and reception in LIN6 and IPv6. For output, in LIN6, a large overhead is incurred because LIN6 needs to search for the mapping of the target packet to perform the embedment process. On the other hand, the overhead is very small for input because LIN6 does not perform the embedment process for input. As the results show, the processing overhead of LIN6 is negligible in comparison with communication time.

V.2 Mapping Table Performance

We measured the processing time of addition, deletion, and search of a random mapping to/from the mapping table while increasing the number of entries in the table in steps of 10 randomly generated entries.

The processing time for each operation increases only slightly with the number of mapping entries(Figure 8) . Although a few monotonous increases occur, these are not a serious problem since a mobile node communicate with a large number of correspondent nodes(about max. 120 nodes/2months). The reason for the gradients is that the mapping table implementation uses a normal hashing algorithm. We can improve the performance by using a better algorithm such as B-tree with the hash.

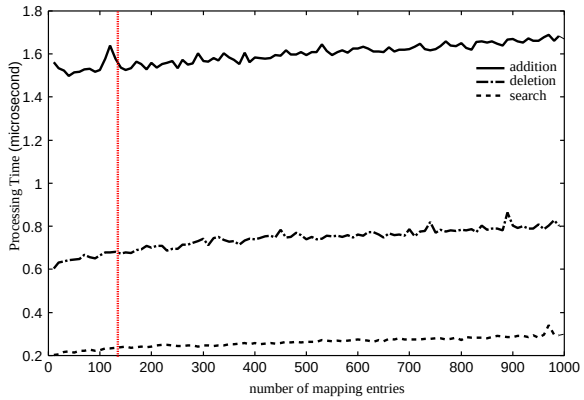

Figure 8: Mapping table processing time

VI. Comparison of LIN6 and Mobile IPv6

LIN6 does not have the problems of Mobile IPv6. First, since LIN6 uses no extension headers of IPv6, it has no header overhead whereas Mobile IPv6 has 48 bytes of header overhead at most. Second, in LIN6, the MAs of a mobile node can be found without location of a node identifier of the mobile node whereas the HA of a mobile node must be located on the subnet specified by the home address of the mobile node. This feature increases fault tolerance. Third, LIN6 is consistent with the IP security protocols because the node identifier that is used above the transport layer is immutable and the node identifier appears in the IPv6 base header.

VII. Futurework

VII.1 Micro Mobility for Smooth Handoff

In wireless communications, mobile nodes connect to the Internet via base stations and change their point of access frequently. A change of access point during active data transmission or reception is called a handoff. During or immediately after a handoff, packet losses may occur due to delayed propagation of location update information[7, 8]. These losses should be minimized in order to avoid a degradation of service quality as handoff become more frequent.

To solve this problem we must design a protocol that provides mobility and handoff support for mobile hosts that change base stations frequently. We will study methods of local level smooth handoff and a hierarchical approach to support wireless network mobility.

VII.2 LIN6 with IPsec

Security issue is a potentially contentious in LIN6's mapping update message. If the mapping update message is attacked by another node, communication packets aren't delivered to the actual LIN6 address, LIN6 nodes can't communicate.

LIN6 needs IP level security such as IPsec[3]. mapping update messages should be authenticated with IPsec AH[6]. When two LIN6 nodes communicate with IPsec, IPsec Security Association should be established using a generalized identifier. When LIN6 node sends a mapping update message to an MA, LIN6 needs to have security mechanisms such as secure DNS update[9, 10].

References

[1] D. B. Johnson and C. Perkins, "Mobility Support in IPv6," draft-ietf-mobileip-ipv6-12.txt, Internet-draft, IETF (2000) (Work in progress).

[2] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC2460, IETF (1998).

[3] S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol," RFC2401, IETF (1998)

[4] R. Hinden, M. O'Dell, and S. Deering, "An IPv6 Aggregatable Global Unicast Address Format," RFC2374, IETF (1998).

[5] Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority, `http://standards.ieee.org/regauth/oui/tutorials/EUI64.html`, IEEE (1997).

[6] S. Kent, "IP Authentication Header," RFC2402, IETF (1998).

[7] Andras G. Valko, "Cellular IP: A New Approach to Internet Host Mobility," *ACM SIGCOMM Computer Communication Review*, vol.29, pp. 50-65, Jan. 1999.

[8] R. Ramjee, T. La Porta, S. Thuel, K. Varadhan, and S.Y. Wang, "HAWAII: A Domain-based Approach for Supporting Mobility in Wide-Area Wireless Networks," *IEEE ICNP 1999*, Oct. 1999.

[9] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound, "Dynamic Updates in the Domain Name System," RFC2136, IETF(1997).

[10] D. Eastlake, "Secure Domain Name System Dynamic Update," RFC2137, IETF 1997.