Multihoming Support based on Mobile Node Protocol LIN6

Arifumi Matsumoto, Kenji Fujikawa, Yasuo Okabe
Graduate School of Informatics,
Kyoto University
Yoshida-Honmachi, Sakyo-ku, Kyoto, JAPAN
arifumi@kuis.kyoto-u.ac.jp

Masataka Ohta
Graduate School of Information Science
and Engineering,
Tokyo Institute of Technology
mohta@necom830.hpcl.titech.ac.jp

Fumio Teraoka, Mitsunobu Kunishi
Graduate School of Science and Technology,
Keio University
tera@tera.ics.keio.ac.jp, kunishi@tokoro-lab.org

Masahiro Ishiyama
Communication Platform Laboratory,
Corporate R&D Center, Toshiba Corporation
masahiro@isl.rdc.toshiba.co.jp

Abstract

We extend the mobile network architecture protocol "LIN6", to support multihoming, and design a new Application Program Interface (API) that enables an application to support multihoming. The LIN6's addressing architecture makes it possible to support what we call end-to-end multihoming without any inconsistency. In end-to-end multihoming, a fault-tolerant connection can be achieved relying not on routers but on the pair of end-nodes only. The new APIs make it easy to write an application that supports multihoming and robust connections. The extension to the LIN6 protocol and the new APIs give the capability of multihoming to the LIN6 protocol.

1 Introduction

Mobile computing, which enables access to the Internet on the street, is getting more popular nowadays, as handy phones become more functional, and as wireless LANs and the Internet spreads. Network protocols that support "mobility" are proposed by some organizations such as Internet Engineering Task Force (IETF). Here mobility means that network connection is stable even when a node moves. However, those protocols have serious problems. In MobileIP, packets have to be carried via a home agent. In MobileIPv6, each packet has a larger header including routing header and home address options. These causes serious amount of connection overhead. Furthermore, these protocols also have security problems.

LIN6(Location Independent Network Architectur for IPv6)[1], which is developed by M. Ishiyama, M. Kunishi and F. Teraoka, is one of the mobile network architectures. It solves these problems by dividing an IP address into two parts, "locator" and "identifier," and by using DNS and "mapping agents" which store locations of mobile nodes.

In this study, we extend the LIN6 protocol and allow a mobile node to have multiple locators. We also design a new Application Program Interface (API) that enables us to write multihoming-ready applications. Instead of conventional router-dependent multihoming, we apply what we call "end-to-end multihoming."[2]

In this paper, we describe extensions to the LIN6 protocol, an API specification, and a few examples of how to use the API in order to write an multihoming-capable applications.

2 Related Works

2.1 IETF MobileIP[3][4]

In MobileIP, each packet to a mobile node has to be carried via a home agent, and this is called "triangular routing". This often leads to serious amount of connection overhead. It takes much more seconds to carry a packet from a source to a destination via a home agent. This is a important problem especially for voice communication system, such as VoIP, which sends and receives a lot of small packets.

As another problem, when a mobile node sends a packet to a corresponding node, some firewalls may consider the packet's source address to be invalid and may drop it. This is also an important problem.

Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT-w'03)
0-7695-1873-7/03 $17.00 © 2003 IEEE

2.2 IETF MobileIPv6[5]

In Mobile IPv6, an IP address of IPv6 is longer than that of IPv4 and so is the case of IP header. In Mobile IPv6, an IP header can be much longer because of Home Address Option(HAO) and Routing Header(RH). 44-byte packet header overhead for every packet can be incurred when Mobile IPv6 nodes communicate with each other. A longer header causes more delay. The packet header length overhead of Mobile IPv6 could be a serious problem in those applications, such as VoIP.

2.3 Multihoming

Almost all the conventional multihoming methods are mainly dependent on routers, which advertise one subnetwork to multiple up-stream routers.[6][7] However, this kind of multihoming support makes the global routing table too large and makes the backbone routers overloaded. This method also spoils the hierarchical structure of IPv6 addressing. Furthermore, as the length of IPv6 address is longer thant that of IPv4, the problem becomes more serious.

3 LIN6 Protocol

In conventional network architectures including IPv4/IPv6, the network address of a node denotes both its identity and its location. In LIN6 architecture, we divide a 128bit-long IPv6 address into two parts. The first half is called "locator" and the second half "identifier". A locator only depicts a location and an identifier only depicts an identity. A LIN6 node can identify a corresponding node by examining only the second half of an IP address. This is independent of the first half, which may be changed when the node moves.

The separation of an IP address also makes it possible to support multihoming without any inconsistency. A LIN6 node located in the multihomed network has multiple global locators. Even if an network trouble occurred on one link, a corresponding LIN6 node detects it and can resume communication by using another locator and another link. In this method, a fault-tolerant connection is achieved relying only on the pair of end-nodes, not on routers. We call this method "end-to-end multihoming."

4 Muitihoming support for LIN6

4.1 Extension to LIN6 Protocol

In LIN6 protocol described above, one node can obtain multiple addresses. However, it cannot either register multiple addresses to the location database or notify a corresponding node of them. We made some addtion to the protocol to register, notify and query multiple addresses. The extented LIN6 protocol allows a LIN6 node and an LA mentioned below to handle multiple locators. This is necessary for LIN6 to support end-to-end multihoming.

The figure below shows the procedures of LIN6 nodes' communication.

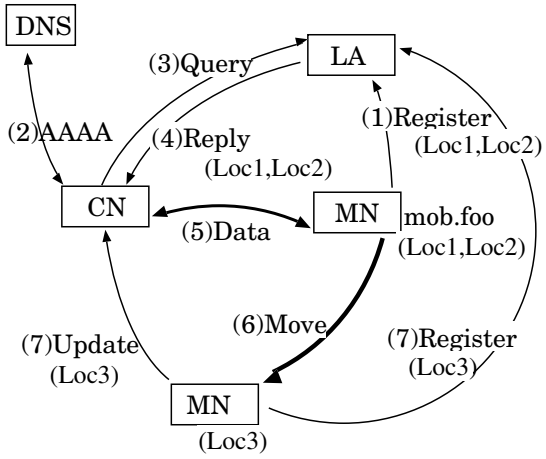

Figure 1. LIN6 Protocol Extension

When a MobileNode(MN) moves to a multihomed network and gets multiple locators(Loc1,Loc2),
1: MN registers all the locators(Loc1, Loc2) to MN's Location Agent(LA) which manages MN's location.
2: A Corresponding Node(CN) which wants to communicates with a MN makes a AAAA-query to a DNS server, and gets an IP address of the MN's LA.
3: CN queries LA where MN is located now.
4: LA replies to CN "MN is located under Loc1 and Loc2."
5: CN communicates with MN directly.
6: MN moves to somewhere else and gets another locator(Loc3).
7: MN updates the location registry in LA and notifies CN of an new location.

As this figure shows, a LIN6 node can register, query and update multiple locaters in the extended LIN6 protocol.

4.2 APIs for Muitihoming support

When a LIN6 node moves and detects its movement (changing of its locator), the node sends a location update message packet to its corresponding nodes and can continue to communicate seamlessly. On the other hand, when a network trouble occurred between two nodes, the connection will be lost, even if one or both nodes are multihomed.

Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT-w'03)
0-7695-1873-7/03 $17.00 © 2003 IEEE

In such a case, a node is expected to detect network troubles by some error packets such as ICMP Host Unreach. The node can use another destination locator and hopefully the connection can be resumed.

These processes is expected to be implemented in an application. Thus, we design new APIs to write an application that supports end-to-end multihoming.

The main newly designed APIs are listed below.

- $socket(AF\_LIN6)$ This API creates a LIN6 socket that identifies a connection from only the second half of the IP address.
- $getaddrinfo2()$ This API returns locators of a specified corresponding node by making a query to the node's Location Agent.
- $getsockopt/setsockopt()$ This API gets/changes the locator of a specified connected socket. When a connection error is detected an application can try another locator by using this API.

In addition to these newly designed APIs, we modified some existing APIs, such as $getaddrinfo()$. This changing made it possible to run existing applications to work well on a LIN6 node. However, as an application cannot get multiple locators by $getaddrinfo()$, it won't benefit from multihoming.

Next, we show the programming example below.

4.3 Example Application

After getting CN's locators by $getaddrinfo2()$, this example applicaton below tries to connect to an acquired address. $getaddrinfo2()$ returns a pointer to a link list of $structaddrinfo2$. If the address family specified in the returned $structaddrinfo2$, there may be multiple locators in it. This application tries to connect to the CN, using each locator.

```
struct addrinfo2 {
  int ai_family; /* PF_XXX */
  int ai_socktype; /* SOCK_XXX */
  struct sockaddr *ai_sockaddr;
  ...
  size_t ai_ntloc;
  /* number of target locator */
  struct lin6_prefix* ai_tloc;
  /* target locators */
}
```

ai_tloc is the pointer to the first entry of the locator array. ai_ntloc is the number of entries in the array.

After the connection establishment this application may catch some error signals caused by ICMP Error Message. This application tries another locator acquired before if available and tries to continue communication.

```
struct addrinfo2 hints,*res,*res0;
struct sockaddr_lin6 slin6;

/* name to address resolution */
hints.ai_family=AF_UNSPEC;
getaddrinfo2("mob.foo","http",
             &hints,&res);

/* look for valid address */
for(res0=res;
    res->ai_next!=NULL;
    res=res->ai_next) {

 sock = socket(res->ai_family,
               res->ai_socktype,
               res->ai_protocol);

 if (res->ai_family==AF_LIN6) {
  memcpy(&slin6,
         res->ai_addr,
         sizeof(slin6));
  for(i=0;i<res->ai_ntloc;i++) {
   /* use each target locator */
   slin6.slin6_locator=res->ai_tloc[i];
   if (connect(sock,
            (struct sockaddr*)&slin6,
            res->ai_addrlen)==0)
    goto connected:
  }
 } else
  connect(sock,res->ai_addr,
          res->ai_addrlen);

connected:
 freeaddrinfo2(res0);
}

void sig_handler(int sig) {
 /* error signal handler */
 if (ai_ntloc>0)
  /* change locator */
  setsockopt(sock,
          IPPROTO_IPV6,
          FOREIGNLOCATOR,
          res->ai_tloc[++i],
          sizeof(struct lin6_prefix));
  ...
}
```


Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT-w'03)
0-7695-1873-7/03 $17.00 © 2003 IEEE

5 Implementation of Muitihoming-ready application

NOTASIP(Nothing Other Than A Simple Internet Phone)[8][9] is an Internet telephone protocol that uses bi-directional UDP stream. We made an NOTASIP application that supports end-to-end multihoming by using these APIs mentioned before. One implementation of multihoming support is to send all the packets to all the locators of a corresponding node. As the figure below shows, when a mobile node is located under multiple wireless stations and has multiple locators, a corresponding node sends each packet to all of the addresses of the mobile node. This can reduce risk of connection down caused by network troubles.

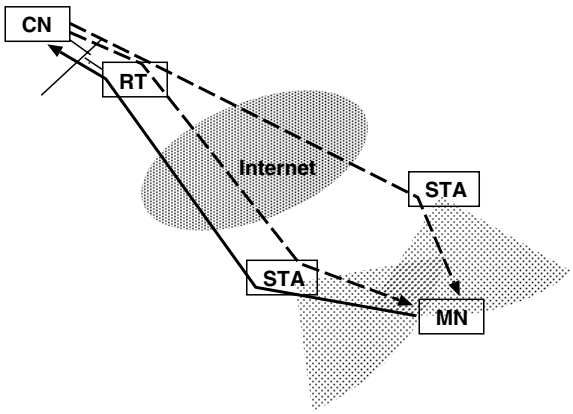

Figure 2. NOTASIP application supporting multiple locators

Another implementation is to use each locator in an order just like round-robin in the DNS system.

It's hard to detect connection trouble when an application uses UDP as a transport layer protocol. However, we can get benefit of multihoming by implementing an application in this way.

6 Conclusion

In this study, we make an extention to LIN6 protocol that enables a LIN6 node to handle multiple locators. We design new APIs to utilize multiple locators in an application. Through this work, LIN6 has become a network architecture that supports mobility and multihoming. As a future work, we make use of the field test network placed in Kyoto prefecture, Japan and commit a lot more experiments.

References

[1] M. Ishiyama, M. Kunishi, K. Uehara, H.Esaki, F. Teraoka, "LINA: A New Approach to Mobility Support in Wide Area Networks," *IEICE Trans. Communications*, Vol E84-B, No 8, Aug 2001, pp.2076-2086.

[2] M. Ohta, "The Architecture of End to End Multihoming," Internet-draft, IETF (Nov 2002), `draft-ohta-e2e-multihoming-03.txt`.

[3] C. E. Perkins, "IP Mobility Support for IPv4," RFC 3344, IETF (Aug 2002).

[4] C. E. Perkins: "Mobile IP: Design Principles and Practices," *Addison-Wesley,* (1998).

[5] D. B. Johnson, C. E. Perkins: "Mobility Support in IPv6," Internet-draft, IETF(2000)(Work in Progress), `draft-ietf-mobileip-ipv6-19.txt`.

[6] T. Bates, Y.Rekhter, "Scalable Support for Multihomed Multi-provider Connectivity," RFC2260, IETF (Jan 1998).

[7] J. Hagino, H.Snyder, "IPv6 Multihoming Support at Site Exit Routers," RFC 3178 IETF (Oct 2001).

[8] Masataka Ohta, Kenji Fujikawa, Takuro Kitagawa, Manolo Sola, Kaz Satoh, "The Simple Internet Phone," Proceedings of INET 2000.

[9] Masataka Ohta, Kenji Fujikawa, "Nothing Other Than A Simple Internet Phone (NOTASIP)," Internet-draft, IETF (Oct 2001), `draft-ohta-notasip-04.txt`, `http://www.notasip.org/`

Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT-w'03)
0-7695-1873-7/03 $17.00 © 2003 IEEE