Development of Mobile Communication Framework based on NTMobile

Kazuma Kamienoo*, Hidekazu Suzuki*, Katsuhiro Naito † and Akira Watanabe*
* Graduate School of Science and Technology, Meijo University, Aichi 468-8502, Japan
Email: 123430013@ccalumni.meijo-u.ac.jp, {hsuzuki, wtnbakr}@meijo-u.ac.jp
†Graduate School of Engineering, Mie University, Mie 514-8507, Japan
Email: naito@elec.mie-u.ac.jp

***Abstract*—We have been proposing "NTMobile" (Network Traversal with Mobility) that realizes secure connectivity as well as mobility enabling the switching of networks during communication in IPv4 and IPv6 networks. In our conventional implementation method adopted for NTMobile, it has been rather difficult for ordinary users to use it because a kernel module is required to be installed in the smartphone, and as a result, it has been an issue how to popularize NTMobile among ordinary users. In this paper, we propose a mobile communication framework that NTMobile functions are all ported in the level of application, and that provides the basis for developing mobile applications suitable for IPv4 and IPv6 networks. We have implemented our proposed method to a commercially available Android smartphone and an iPhone, and verified the full connectivity and mobility.**

***Index Terms*—Mobility, NAT Traversal, IPv4 and IPv6 networks.**

I. Introduction

With the popularization of high-efficiency mobile terminals and the development of wireless technologies, demand for Internet services for mobile terminals and for mobile applications has been rapidly increasing. On the other hand, exhaustion of global IPv4 addresses has become a serious problem in the present Internet, and thus, it has become a common practice to establish private networks in IPv4 networks by introducing NAT (Network Address Translation). In the environment in which NAT is introduced, however, there occurs a so-called "NAT traversal problem" in which connectivity from a node on the global network side to another node on the private network side cannot be secured, which has thus become a factor of spoiling the fundamental idea of the Internet that should ensure end-to-end connectivity. Meanwhile, the Internet is presently in the transitional period from IPv4 to IPv6, IPv4 and IPv6 networks is gradually spreading, despite the fact that they have no compatibility with each other. Under such circumstances, it is necessary to secure connectivity to mobile terminals in the NAT environment as well as in IPv4 and IPv6 networks in order to develop applications that enable direct communication between mobile nodes.

In the meantime, a number of wireless interfaces such as 3G, Wi-Fi and WiMAX are normally equipped in smartphones, and it is possible to make communication by changing interfaces as required. However, because communication is identified based on IP addresses assigned to individual interfaces in the node in IP networks, communication cannot be maintained if and when an IP addresses changes due to the switching of interfaces or networks. The technology to solve this kind of problem is called "IP mobility technology", and various technologies have been proposed thus far [1]–[5]. But, it is difficult for ordinary users to use them with Android or iPhone, because some modification to OS or kernel is required.

In [3], SIP mobility is proposed where mobility is realized based on an application layer using SIP (Session Initiation Protocol). In the case of SIP mobility, the application makes communication based on SIP, and even when networks are switched, communication is maintained on the strength of SIP's session continuation function. In SIP mobility, since there is no need to install any specific module to OS or kernel, communication can be maintained easily even with commercially available with mobile terminals. However, because SIP mobility assumes communication on UDP alone, there exists a problem that TCP session cannot be maintained. Although All-SIP mobility has been proposed as a technology that has solved the above said problem, however it is difficult to use it commercially available, because it requires installation of a virtual interface or an SMC (Session and Mobility Controller) [4].

We have been proposing "NTMobile" (Network Traversal with Mobility) that realizes secure connectivity and mobility which enabling the switching of networks during communication in IPv4 and IPv6 networks, at the same time [6]–[8]. As NTMobile has the function of NAT traversal technology, it can secure connectivity to mobile nodes behind NAT in IPv4 and IPv6 networks, without any modification to NAT. NTMobile has already been implemented to Android smartphones, and its effectiveness in IPv4 and IPv6 networks has already been verified. Yet, in the case of our conventional NTMobile implementation it is rather difficult for ordinary users to use it with ease, because a kernel module and a daemon program are required to be installed to the smartphone.

In this paper, we propose a mobile communication framework that realizes secure connectivity and mobility in IPv4 and IPv6 networks with mobile terminals carried by ordinary users, by realizing the function of NTMobile on the level of application. Mobile application developers are easily able to realize applications that make direct communication between

978-1-4799-2231-4/14/$31.00 ©2014 IEEE

mobile nodes, by using our proposed. We have also developed a prototype of our proposed method and made an evaluation of the handover processing, by using an Android smartphone and an iPhone.

We are going to overview our NTMobile in Chapter II, describe the outline and implementation of our proposed method in Chapter III, the results of our performance evaluation and issues to consider hereafter in Chapter IV and the summary in Chapter V.

II. NTMobile

A. Outline

In NTMobile, virtual IP addresses are assigned to NTMobile nodes (NTM nodes), and the application makes a connection based on virtual IPv4 addresses or virtual IPv6 addresses. Packets based on virtual IP addresses are transmitted by a UDP tunnel established between NTM nodes. As the UDP tunnel is established on an end-to-end basis except for certain specific cases, NTM nodes can make tunnel communication through the most appropriate route. Using the above-mentioned method, applications can make communication freely without being affected by the switching of networks or by the existence of NAT on the communication route, and by the difference of the real IPv4 and real IPv6 networks.

The overview of NTMobile system is shown in Fig. 1. NTMobile consists of NTM nodes, DC (Direction Coordinator), and RS (Relay Server). DC and RS are placed in the Dual Stack Network, and multiple units of them can be set depending on the size of the network.

- NTM node
  NTM node has two kinds of addresses; namely, one is a real IP address assigned by the real network and the other a virtual IP address assigned by DC. The virtual IP address remains unchanged even if NTM node switches its networks. To the NTM node, virtual IP addresses of both IPv4 and IPv6 are assigned, and the application makes communication based on either of the virtual IPv4 address or the virtual IPv6 address.
- DC (Direction Coordinator)
  DC is a coordinator to manage the assignment of virtual IP addresses and to give instructions to the NTM node to establish tunnels. The virtual IP address to be assigned to the NTM node is a unique address and each DC undertakes assignments in a way that no duplication occurs from the address space allocated to itself. DC also is managing address information of the NTM node by using database. In the database of DC, such information as real IPv4 addresses, real IPv6 addresses, virtual IPv4 addresses, virtual IPv6 addresses, and real IPv4 addresses outside NAT, is registered.
- RS (Relay Server)
  RS is a server to relay communications under certain specific circumstances; e.g., when communication is made between NTM nodes located behind different NATs, or communication is made with a general node which does not have the function of NTMobile. When NTM nodes behind different NATs are to make communication with each other, it is possible to switch to the end-to-end tunnel route by conducting route optimization. Also, when communication is made between different address families like the case of NTM node A and NTM node B, as illustrated in Fig. 1, tunnels are established with RS set in the Dual Stack Network, and communication is made via RS. Multiple units of RS can be set on the Internet, and the most suitable RS is selected at the time of creating a tunnel by taking account of the relay load and the route redundancy.

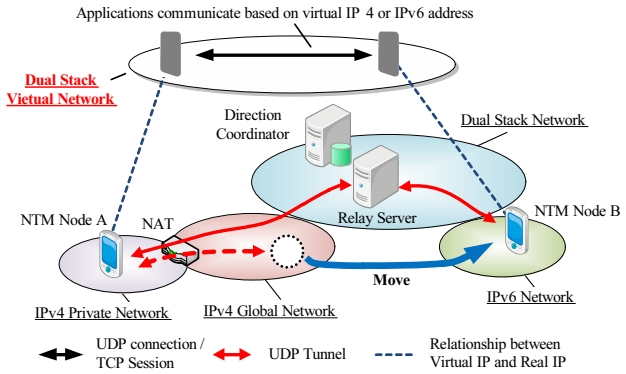


Fig. 1. Overview of NTMobile.

Based on the assumption that DC and each node have a reliable relationship, each message used by NTMobile is encrypted by an encryption key, and MAC (Message Authentication Code) is added. Encryption key is delivered from DC at the time of tunnel creation.

B. Implementation of NTM node and Issues to Consider

NTMobile has already been implemented to Linux PC and Android smartphone, and its effectiveness in IPv4 and IPv6 networks has already been verified. The NTM node functions when a daemon program (NTM daemon) which performs tunnel creation process, a kernel module which performs encapsulation and encryption process of application packets with real IP addresses, and a virtual interface, are implemented. The NTM node starts tunnel creation process when its kernel module has hooked a request for name resolution, and returns a name resolution response to the application, by describing the virtual IP address of the correspondent node. Through this process, the application recognizes the virtual IP address as the IP address of the correspondent node and starts communication based on the virtual IP address. Communication packets based on the virtual IP address are encapsulated and encrypted by the kernel module and transmitted to the correspondent node through the UDP tunnel established between NTM nodes. NTM node, when receiving encapsulated packets, decrypts and decapsulates them by its kernel module, and receives extracted virtual IP packets from the virtual interface. Through the above-mentioned process, the application of NTM nodes

performs communication based on virtual IP address.

In order to install a kernel module and an NTM daemon to an Android smartphone, root authorization needs to be obtained and the kernel needs to be reconstructed. Thus, it has been difficult for ordinary users to use NTMobile with ease. Another problem has been that it cannot used in iPhones which do not have a Linux kernel. So the question has been how to popularize the system among ordinary users.

III. Mobile Communication Framework

A. Outline

In this paper, we propose a mobile communication framework that realizes full connectivity and mobility in IPv4 and IPv6 networks by using commercially available Android smartphones or iPhones, by implementing the NTM node function as the framework on the level of application.

The outline of the mobile communication framework is shown in Fig. 2. NTMobile is applied in order to secure mutual connectivity in IPv4 and IPv6 networks, to solve NAT traversal problem, and to realize mobility. For those purposes, we set multiple units of DC and RS in real networks, like the case of the NTMobile network configuration as explained in Chapter II. The NTM node is based on commercially available devices such as Android smartphones and iPhones equipped with mobile applications. In our proposed method, unlike the case of conventional NTM nodes, we implement the mobile communication framework with the function of NTM node to the level of application. With this method, each user is able to make mobile communication based on NTMobile, by merely installing a mobile application to his or her own mobile terminal.

The module configuration of a mobile communication framework is shown in Fig. 3. The mobile communication framework consists of such modules as a negotiation module which performs tunnel creation process, a packet manipulation module which performs encapsulation and encryption process, and a handover module which detects changes in the state of network connections based on the communication management function provided by OS. In addition, the mobile communication framework provides the application with virtual socket API so as to conduct communication by virtual IP addresses. The application makes communication using the API provided by the mobile communication framework, instead of the API provided by OS. Through this process, the detection of name resolution process and encapsulation/encryption process of packets based on virtual IP addresses which has been conducted by a kernel module in the conventional NTM node, are realized on the level of application. Meanwhile, as the virtual socket API provided by the mobile communication framework has compatibility with ordinary socket APIs like BSD (Berkeley Software Distribution) socket, applications can utilize it in the same procedures as those of the ordinary socket API.

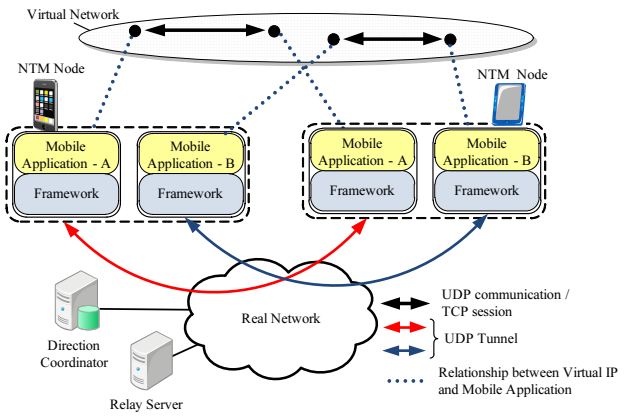


Fig. 2. Overview of Mobile Communication Framework.

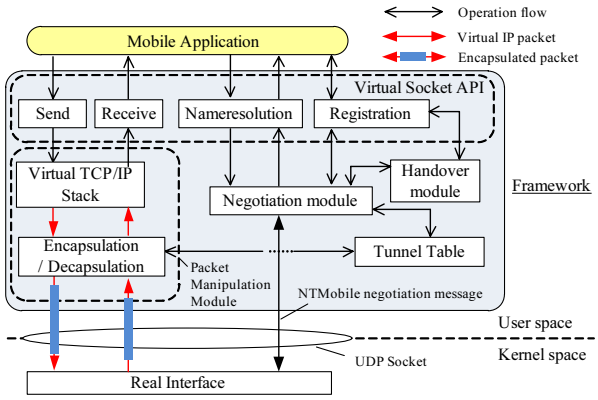


Fig. 3. Module configuration of Mobile Communication Framework.

B. Communication process

Fig. 4 shows the situation where NTM node MN (Mobile Node) which is connected to the IPv4 global network makes UDP communication of the virtual IPv6 based on a mobile application using the mobile communication framework, with NTM node CN (Correspondent Node). Hereinafter, we call FQDN at the node X as "$\mathrm{FQDN_X}$", virtual IPv6 address as "$\mathrm{VIP6_X}$", real IPv4 address as "$\mathrm{RIP4_X}$", real IPv6 address as "$\mathrm{RIP6_X}$". Also, we call Path ID to identify the tunnel established between MN and CN as "$\mathrm{PID_{MN-CN}}$".

1) Registration Process
MN's and CN's applications undertake registration process for NTMobile at the time of start-up, by calling up Registration API. Through this process, the real IP addresses which MN and CN obtained from the network are registered in DC. It is noted that the registration process is the only specific process undertaken by applications using the mobile communication framework, and all processes thereafter are performed in the same manner as that of the case of using ordinary socket API. In the case of our example described in Fig. 4, MN's and CN's applications generate virtual sockets based on the virtual socket connection API and wait

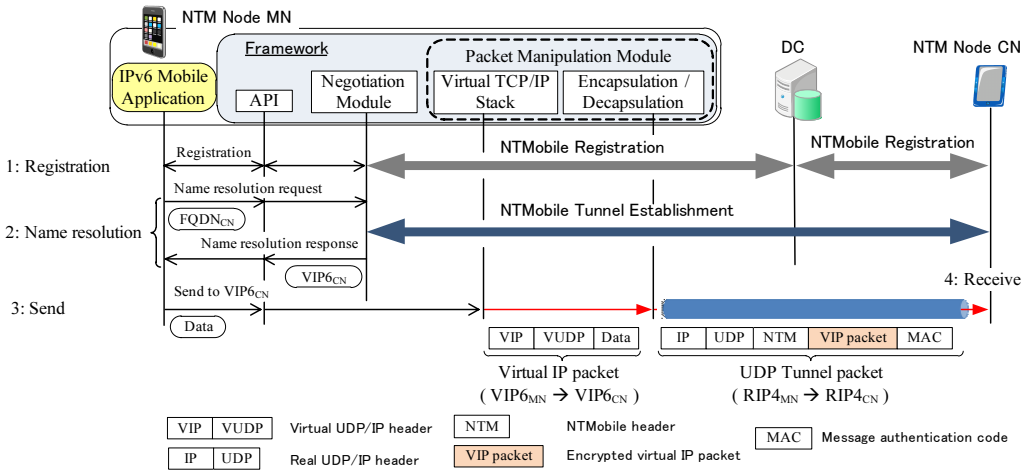


Fig. 4. Communication procedure of Mobile Communication Framework.

for communication from other terminals.

2) Name Resolution Process
MN's application executes the name resolution processing of $FQDN_{CN}$, by using the virtual socket API before starting communication to CN. On that occasion, the tunnel establishment process is executed by the negotiation module and a tunnel between MN and CN is established. After the tunnel establishment, MN's application obtains VIP_{CN} as the result of the name resolution processing.

3) Sending Process
After completing the name resolution process, MN's application sends data towards $VIP6_{CN}$ by using the virtual socket API. When the virtual socket API is called up, a virtual IP packet addressed to $VIP6_{CN}$ is created from $VIP6_{MN}$ by the packet manipulation module. Thereafter, the packet manipulation module searches the tunnel table by using the virtual IP address $VIP6_{CN}$ as the search key, and encapsulates the virtual IP packet in accordance with the entry. At the time of encapsulation, an NTM header describing PID_{MN-CN} is attached to the virtual IP packet in order to perform encryption and MAC addition. Then, the encapsulated packet is transmitted to $RIP4_{CN}$, which is the other end of the established tunnel. At the time of transmitting the packet, since the packet is sent from the UDP socket generated by the socket API provided by OS, the virtual IP packet sent by the application to $VIP6_{CN}$ is encapsulated by the UDP packet based on the real IP address and transmitted to $RIP4_{CN}$.

4) Receiving Process
The packet manipulation module of CN's application, upon receiving an encapsulated packet, searches the tunnel table by PID_{MN-CN} described in the NTM header of the encapsulated packet as the search key, and performs decryption and decapsulation in accordance with the entry. The packet manipulation module hands over the payload portion of the virtual IP packet to the virtual socket to which the destination port number of the extracted virtual IP packet is assigned. Through this process, CN's application receives data sent by MN's application through the virtual socket created by the virtual socket API.

By undergoing all the above processes, MN's and CN's applications are able to make communication based on the virtual IP addresses.

C. Handover Process

The handover module executes the handover process at the time when networks are switched as a result of mobile node relocation or Wi-Fi's on/off operation. The handover module calls the negotiation module and registers in DC the real IP address obtained from the network in the same procedure as that adopted at the time of an application start-up. In this way, reachability to NTM Node is secured.

In the meantime, the negotiation module reestablishes a tunnel towards CN in the same procedure as that adopted at the time of a communication start-up. At the time of tunnel reestablishment, DC indicates the most suitable tunnel route, corresponding to the network by which MN and CN are connected, and a tunnel is reestablished between MN and CN. Because the applications of MN and CN are making communication based on the virtual IP addresses, the communication can be kept without being affected by the change of real IP addresses.

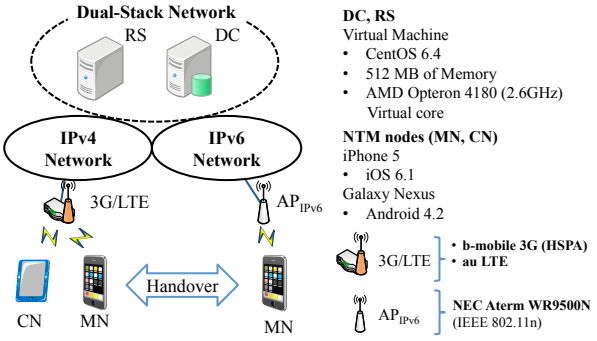
IV. Evaluation and Issues for Consideration

A. Performance evaluation

We conducted a behavior verification and the performance evaluation of handover process by implementing a mobile application using a prototype of the mobile communication framework to an iPhone as well as to an Android smartphone. Meanwhile, to the prototype virtual TCP/IP stack, we implemented UDP functions this time.

TABLE I
THE MEASUREMENT PATTERN

	Mobile Device		Access Network	
	MN	CN	MN (before/after the handover)	CN
Case 1-1	iPhone 5	Galaxy Nexus	LTE (IPv4) ⇒ Wi-Fi (IPv6)	3G (IPv4)
Case 1-2	Galaxy Nexus	iPhone 5	3G (IPv4) ⇒ Wi-Fi (IPv6)	LTE (IPv4)
Case 2-1	iPhone 5	Galaxy Nexus	Wi-Fi (IPv6) ⇒ LTE (IPv4)	3G (IPv4)
Case 2-2	Galaxy Nexus	iPhone 5	Wi-Fi (IPv6) ⇒ 3G (IPv4)	LTE (IPv4)



	DC-RS	Galaxy Nexus				iPhone 5			
		b-mobile 3G		Wi-Fi		au LTE		Wi-Fi	
		DC	RS	DC	RS	DC	RS	DC	RS
RTT [ms]	0.3	446.0	508.3	10.3	10.4	279.3	172.9	31.9	33.4

Fig. 5. Evaluation environment.

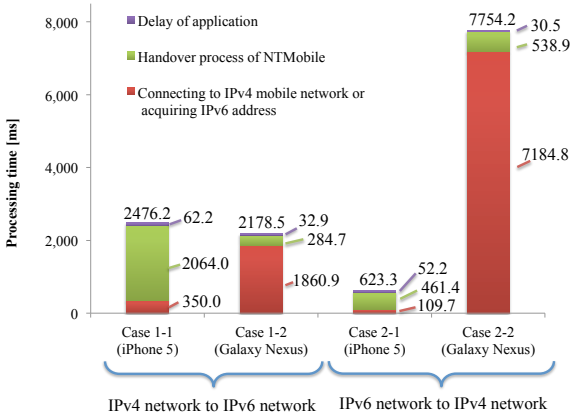


Fig. 6. Results of suspended time by handover.

The measurement environment and the average RTT (Round-Trip Time) between certain specific types of devices are shown in Fig. 5. DC and RS are established with virtual machines, and Apple iPhone 5 and Samsung Galaxy Nexus are used for MN and CN. AP_{IPv6} was set as an ordinary access point by inactivating its router function. It should be noted here that with the Galaxy Nexus which we used this time, forcible disconnection from AP occurred at the time when we failed the operation of obtaining IPv4 address based on DHCP (Dynamic Host Configuration Protocol) and as a result, we could not get a connection with IPv6 network. Because of such an unexpected problem, we statically established 0.0.0.0 as the IPv4 address in Wi-Fi interface of Galaxy Nexus when connecting with AP_{IPv6}. At the time when iPhone 5 and Galaxy Nexus were connected to AP_{IPv6}, connection was conducted by IEEE 802.11n, and WPA/WPA2-PSK (AES) was used as encryption and authentication functions. And, when Wi-Fi is in an inactivated state, iPhone 5 is connected to IPv4 mobile network based au LTE and Galaxy Nexus is connected to IPv4 mobile network based b-mobile 3G.

We used iPhone 5 and Galaxy Nexus as MN and CN and measured communication interruption time which occurred as a result of the switching of connected networks during communication between MN and CN. We kept CN being connected to IPv4 mobile network all the time, and we manually switched MN's connected in the follows Tab. I.

For the purpose of measurement, we created an application which sends UDP packets at an interval of 50 ms, and made communication from MN to CN. In order to measure the time required for handover of NTMobile, we added a program to output a time stamp for each processing, and calculated the time required for handover process based on the time differences and capturing results of communication packets.

The average values of 10 measurements of communication interruption time associated with the handover processing are shown in Fig. 6. Communication was interrupted for 2,476.2 ms in Case 1-1 and for 2,178.9 ms in Case 1-2. At this occasion, disconnection from IPv4 mobile network was conducted in iPhone 5 and Galaxy Nexus after authentication and connection to AP_{IPv6} were completed, and the above-said communication interruption time occurred. Acquisition process of IPv6 address by IPv6 stateless address autoconfiguration [9] was performed after completing connection to AP_{IPv6}, and for that operation, Case 1-1 required 350.0 ms and Case 1-2 required 1,860.9 ms. These processing times include process delays that occurred due to the times which the mobile communication framework needed before it detected the handover. After acquiring IPv6 address, handover process by NTMobile was performed, and for that operation, Case 1-1 required 2,064.0 ms and Case 1-2 required 284.7 ms. Then, communication by the application resumed after 62.3 ms in Case 1-1, and after 32.9 ms in Case 1-2.

In Case 2-1 communication was interrupted for 623.3 ms, and in Case 2-2 communication was interrupted for 7,754.2 ms. In Case 2-1, 109.7 ms was required from the time Wi-Fi was inactivated before the mobile network was connected, and in Case 2-2, the time required for the same operation was 7,184.8 ms, which accounted for 92% of the entire communication interruption time. These processing times include

process delays that occurred due to the times which the handover module needed before it detected the handover. After completing connection to the mobile network, the handover process by NTMobile was performed, and for that operation, Case 2-1 required 461.4 ms and Case 2-2 required 538.9 ms. Then, communication by the application resumed after 52.2 ms in Case 2-1, and after 30.5 ms in Case 2-2.

Case 1-2 required 5 more times than Case 1-1 for the processing to acquire IPv6 address. The difference in the processing times is thought to have occurred due to the difference in the implementation modes of IPv6 of iOS and Android, because in the case of Linux, it is set to wait for about 1 second when executing DAD (Duplicate Address Detection), which is programmed to be performed at the time when IPv6 address is obtained [10].

Case 1-1 required 7 more times than 1-2 for the handover processing by NTMobile. In Case 1-1, there was a waiting time of about 900 ms before iPhone 5 sent NS (Neighbor Solicitation) targeting at DC and RS, while the processing by NTMobile was completed in about 230 ms. As there is no waiting time occurring in Case 1-2 before Galaxy Nexus sent NS, the delay is thought to have occurred due to the difference in the implementation modes of IPv6 communication processes of iOS and Android OS.

B. Issues to Consider Hereafter

From the measurement results, it was found out that while the processing by NTMobile at the time of handover takes only less than 538.9 ms, a delay of more than 1 second occurs due to the difference in the implementation modes of IPv6 communication processes of different kinds of smartphones. It was also found out that in the case of Galaxy Nexus, it takes more than 7 seconds before the connection to IPv4 mobile network is made after Wi-Fi is inactivated, which time period accounts for 92% of the entire communication interruption time. When we assume the usage for voice and animation that require real time communication, it is desirable to make the communication suspended time as short as possible, and therefore, it is required to reduce the delay time that occurs in IPv6 communication processing and in the processing for connection to the mobile network.

By utilizing the mobile communication framework based on our proposed method, we can offer mobile communication, such as communication between IPv4 and IPv6 networks and mobility, to commercially available mobile terminals. On the other hand, because it is necessary to make communication by using the virtual socket API if the application is to make mobile communication, the communication protocols that can be utilized by the application are limited to such communication protocols that are offered by virtual TCP/IP stacks. For that reason, the application cannot make other communications than UDP as long as it relies upon the prototype implementation mode outlined in Section IV-A above. In lwIP (Lightweight TCP/IP stack)[1] and uIP (micro IP)[2], handy implementation modes for TCP/IP in Userspace are made open to the public, and they can be used not only for UDP but also for the processing of TCP and ICMP. Thus, we think that we will also be able to offer other communication protocols such as TCP and ICPM in the virtual TCP/IP stack, by taking advantage of these implementation modes hereafter.

[1] http://savannah.nongnu.org/projects/lwip/

[2] http://dunkels.com/adam/

V. Summary

In this paper, we proposed a mobile communication framework using NTMobile that realizes full connectivity and mobility in the IPv4 and IPv6 networks. By using the mobile communication framework, application developers are able to easily develop applications with high-grade real-time communication as well as applications that can make direct communication between mobile terminals without the necessity of preparing for large-scale servers. Meanwhile, we implemented a mobile application using a prototype of the mobile communication framework to a commercially available Android smartphone as well as to an iPhone, and conducted a behavior verification and performance evaluation of the system. As the result, we could verify that the securing of connectivity and mobility are feasible without being affected by the existence of NAT or the difference of IPv4/IPv6 networks. It was confirmed that because of the reason that communication interruption times occur at the time of handover owing to the network connection processing, it is necessary to shorten this process.

VI. Acknowledge

This research was supported by Strategic Information and Communications R&D Promotion Programme (SCOPE) of the Ministry of Internal Affairs and Communications, Japan.

References

[1] D. Le, X. Fu, and D. Hogrefe, "A Review of Mobility Support Paradigms for the Internet," *IEEE Communications Surveys*, vol. 8, no. 1, pp. 38–51, 2006.

[2] H. Soliman, "Mobile IPv6 Support for Dual Stack Hosts and Routers," *RFC 5555,IETF*, 2009.

[3] H. Schulzrinne and E. Wedlund, "Application-layer mobility using SIP," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 4, no. 3, pp. 47–57, 2000.

[4] H. Miyajima, L. Zhang, H. Hayashi, and T. Fujii, "An implementation of enhanced all-sip mobility," *Proc. of IEEE PIMRC2008*, pp. 1–6, 2008.

[5] T. Okoshi, M. Mochizuki, Y. Tobe, and H. Tokuda, "Mobile-socket:session layer continuous operation support for java applications," *Transactions of Information Processing Society of Japan*, vol. 41, no. 2, pp. 222–234, 2000.

[6] H. Suzuki, K. Kamienoo, T. Mizutani, T. Nishio, K. Naito, and A. Watanabe, "Design and Implementation of Establishment Method of Connectivity on NTMobile," *Transactions of Information Processing Society of Japan*, vol. 54, no. 1, pp. 367–379, 2013.

[7] K. Naito, T. Nishio, K. Mori, H. Kobayashi, K. Kamienoo, H. Suzuki, and A. Watanabe, "Proposal of seamless ip mobility schemes: Network traversal with mobility (ntmobile)," *IEEE Global Communications Conference (GLOBECOM) 2012*, pp. 2572–2577, 2012.

[8] K. Kamienoo, H. Suzuki, K. Naito, and A. Watanabe, "Implementation and evaluation of ntmobile with android smartphones in ipv4/ipv6 networks," *IEEE 1st Global Conference on Consumer Electronics (GCCE) 2012*, pp. 125–129, 2012.

[9] S. Thomson and T. Narten, "IPv6 Stateless Address Autoconfiguration," *RFC4862,IETF*, 2007.

[10] P. Pongpaibool, P. Sotthivirat, S. Kitisin, and C. Srisathapornphat, "Fast duplicate address detection for mobile ipv6," *Proceedings of the 15th IEEE International Conference on Networks, ICON 2007*, pp. 224–229, 2007.