Realization and Evaluation of Java Wrapper that calls the End-to-End Communication Library

Kazuki Shimizu†, Hidekazu Suzuki†, Katsuhiro Naito‡, Akira Watanabe†
†Graduate School of Science and Technology, Meijo University, Aichi 468-8502, Japan
‡Faculty of Information Science, Aichi Institute of Technology, Aichi 470-0392, Japan
Email: kazuki.shimizu@wata-lab.meijo-u.ac.jp,
hsuzuki@meijo-u.ac.jp, naito@pluslab.org, wtnbakr@meijo-u.ac.jp

I. Introduction

With the spread of mobile communication nodes such as smartphones and wireless communication technologies, the demand for the use of networks has been increasing. In the case of IPv4 networks, exhaustion of global IP addresses has been a serious problem. As a short-term solution for this problem, the use of NAT is quite common. However, there exists a problem which associated with NAT as well. Namely, we cannot make communication from the global network side to the private network side behind NAT. As a long-term solution for the IPv4 address exhaustion problem, it is necessary to migrate from IPv4 to IPv6 network environment. However, there is no compatibility between these networks, and thus, IPv6 is not widely spread at all. Because of the above-said problem, the mixed environment of IPv4 and IPv6 networks seems to last for a long period of time. Accordingly, "connectivity" is required so that communication can be maintained regardless of the connected network environment. Furthermore, the communication node cannot continue its communication when it moves to another network, owing to the change of its IP Address. Therefore, "mobility" is required so that the node can continue communication even if the network is switched to another network.

In this paper, we define the communication that realizes "connectivity" and "mobility" at the same time "end-to-end communication". As a technology that realizes end-to-end communication, "NTMobile" (Network Traversal with Mobility)[1] has been proposed. NTMobile is a system which allocates a unique virtual IP address to each node, and every communication packet is encapsuled by the real IP address. There is a communication library named "NTMobile Framework"[2], which is one of the implementation models of NTMobile, realized in the application layer. Applications can realize end-to-end communication by using this communication library. Yet, there is a problem that the development environment for applications is limited to C-language because the NTMobile Framework is implemented in C-language. Therefore, in this paper, we have studied "Java wrapper" which calls the NTMobile Framework from Java. We have realized 2 types of Java wrappers, and confirmed that Java applications can call the NTMobile Framework. We found out that there was almost no difference in the performance between the 2 types of Java wrappers, although they had advantages and disadvantages, depending on the situation.

II. NTMobile

A. Outline

NTMobile is composed of "NTM Node" (NTMobile Node) which has the functions of NTMobile, "DC" (Direction Coordinator) which manages virtual IP addresses and location information of NTM Nodes and directs NTM Nodes to create a UDP tunnel, and "RS" (Relay Server) which relays packets when NTM Node cannot make a direct route between NTM Nodes. It is possible to set DCs and RSs in a distributed manner on the Internet.

DC allocates a location-independent virtual IP address to each NTM Node, and an application makes communication based on the virtual IP address. Every packet based on the virtual IP address is encapsulated by the real IP address of NTM Node and sent to the network. Each NTM Node behind NAT maintains communication path with DC so as to receive control packets from DC all the time. The virtual IP address does not change to whichever network the NTM Node moves to. Thus, applications are not affected by the change in the real IP addresses due to relocations, and mobility can be maintained.

B. NTMobile Framework

The NTMobile Framework is one of the implementation models of NTMobile which provides users as an application library. The NTMobile Framework is implemented in C-language, and it provides NTM socket API having compatibility with BSD socket API. Applications can communicate with NTMobile by calling the NTM socket API, and its usage is almost the same as that of BSD socket API.

III. Proposed Method

In this paper, we propose 2 types of Java wrappers, and they are herein named "Inheritance-type Java Wrapper" and "Factory-type Java Wrapper". Both of them use "JNA" (Java Native Access), which is open source software, to call the NTMobile Framework from Java. Fig. 1 shows the structure of the Java wrappers.

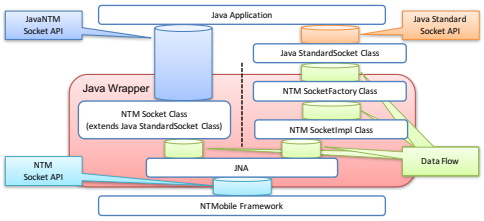


Fig. 1. Structure of Java Wrappers

TABLE I
MEASUREMENT RESULTS OF THE PROCESSING TIME OF INHERITANCE-TYPE JAVA WRAPPER

Measurement Point	Sending Time[ms]	Receiving Time[ms]
Java Wrapper	0.83	0.17
NTMobile Framework	0.41	1.20
Total	1.24	1.37

A. Inheritance-type Java Wrapper

The Inheritance-type Java Wrapper calls the NTM socket API by using the Java socket API from wrapper classes which inherit the Java standard socket classes. When a Java application calls API defined in the wrapper classes, Java wrapper eliminates all differences between C-language and Java. After that, the NTM socket API is called.

B. Factory-type Java Wrapper

The Factory-type Java Wrapper redefines the Java standard socket API so as to call the NTM socket API by setting socket implementation factory to the application. In this method, the application cannot use the standard communication prepared in Java, after the socket implementation factory is set. However, the application can communicate with NTMobile by using the Java standard socket API itself.

IV. EVALUATION

A. Measurement of Performance

We implemented both Inheritance-type and Factory-type Java Wrapper, and applied them to Java application which sends and receives messages by way of UDP. We constructed DC and 2 NTM Nodes by virtual machines in the host machine and connected them to the same network. After that, we have verified the operation and measured the processing time.

TABLE I shows the measurement result of Inheritance-type Java Wrapper, and TABLE II shows that of Factory-type Java Wrapper. They are the average of 100 times of measurement. From these results, we found out that there was almost no difference in the processing time between the 2 types of Java wrappers.

B. Comparison

TABLE III shows advantages and disadvantages of both types. Our comparison was focused on the following 3 points.

1) Easiness of Development
2) Extendability

TABLE II
MEASUREMENT RESULTS OF THE PROCESSING TIME OF FACTORY-TYPE JAVA WRAPPER

Measurement Point	Sending Time[ms]	Receiving Time[ms]
Java Wrapper	0.87	0.19
NTMobile Framework	0.43	1.22
Total	1.30	1.41

TABLE III
COMPARISONOF 2 TYPES OF JAVA WRAPPERS

	Item (1)	Item (2)	Item (3)
Inheritance-type Java Wrapper	○	○	△
Factory-type Java Wrapper	○	×	○

3) Scalability

In the comparison of Item (1), the application can communicate with NTMobile by the Java standard socket API when it uses Factory-type Java Wrapper. On the other hand, Inheritance-type Java Wrapper is required to use original socket classes when the application wants to communicate with NTMobile. However, the usage of API is the same as that of the Java standard socket API. Thus, as for the easiness of development, comparison result was the same.

In the comparison of Item (2), the application can select between the normal communication and the NTMobile communication when it uses Inheritance-type Java Wrapper. On the other hand, the application automatically selects the NTMobile communication in the case of Factory-type Java Wrapper. Thus, for the extendability, Inheritance-type Java Wrapper is superior to Factory-type Java Wrapper.

In the comparison of Item (3), the application can change the socket communication to the NTMobile communication by setting socket implementation factory to the application in the case of Factory-type Java Wrapper. Thus, for the scalability, Factory-type Java Wrapper is superior to Inheritance-type Java Wrapper.

V. CONCLUSION

In this paper, we discussed the Java wrapper that enables Java to call the NTMobile Framework which was limited to C-language applications. We have proposed 2 types of Java wrappers, and realized both of Java wrappers and verified their operation. We found that there was almost no difference in the performance between the 2 types of Java wrappers, although there were advantages and disadvantages, depending on the situation. Hereafter, we plan to study other types of wrappers with different programming languages.

REFERENCES

[1] K. Naito, K. Kamienoo, T. Nishio, H. Suzuki, A. Watanabe, K. Mori, and H. Kobayashi. Proposal of Seamless IP mobility schemes:Network traversal with Mobility(NTMobile). *IEEE Global Communications Conference(GLOBECOM)2012*, pp. 2572–2577, 2012.

[2] K. Naito, K. Kamienoo, H. Suzuki, A. Watanabe, K. Mori, and K. Kobayashi. End-to-end IP mobility platform in application layer for iOS and Android OS. In *Proc. of IEEE CCNC*, 2014.

Realization and Evaluation of Java Wrapper that Calls the End-to-End Communication Library

Kazuki Shimizu
Meijo University, Japan

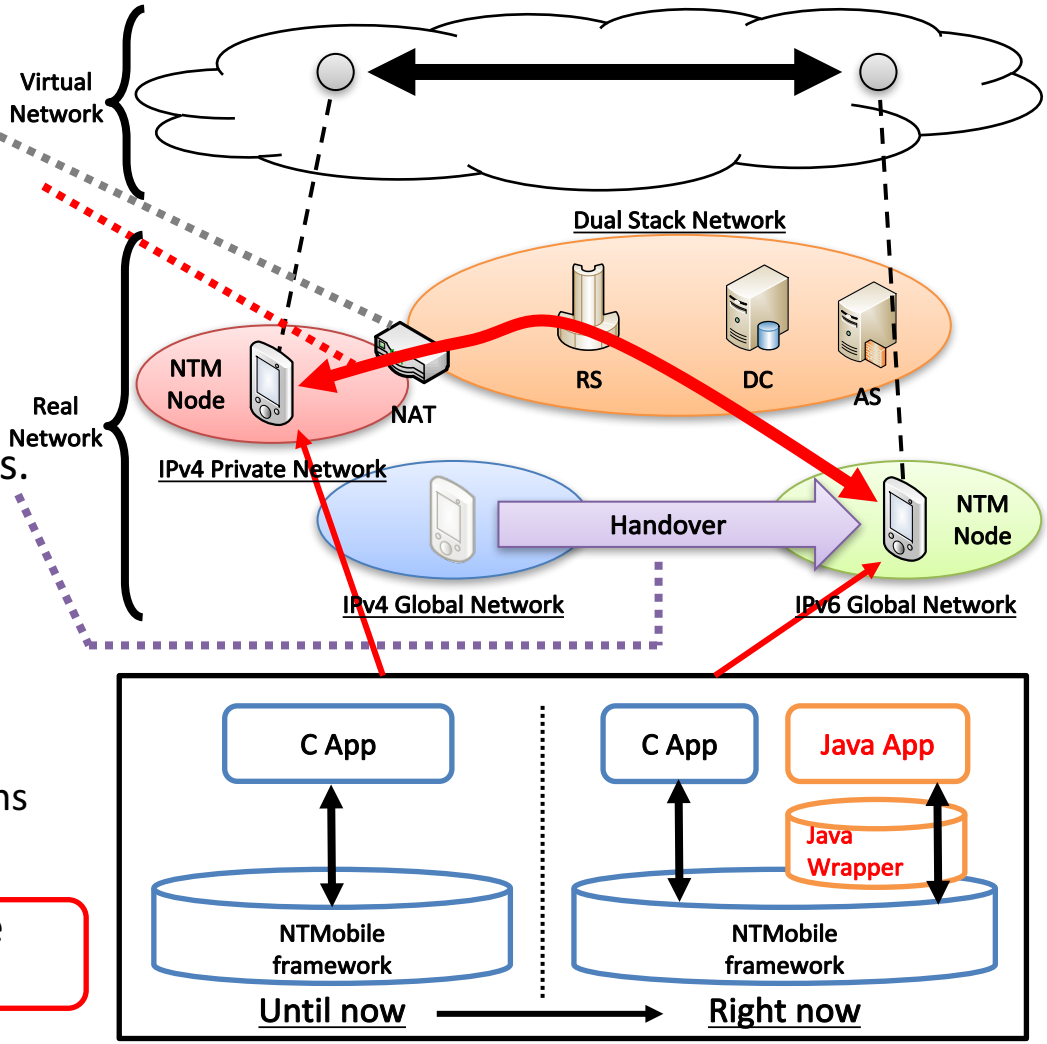
Watanabe Lab.

- **Restrictions of networks**
  - NAT traversal problem
  - Incompatibility between IPv4/v6
  - Mobility

  ⇩ Solution of restrictions
- **NTMobile framework**
  - App library of NTMobile functions.
  - Written in C-language.
    - It can be called only by C apps.

  ⇩
- **Java Wrapper**
  - Java wrapper enables Java applications to use NTMobile framework.

  ⇩ NTMobile has become easier to use.



Realization and Evaluation of Java Wrapper that Calls the End-to-End Communication Library

Kazuki Shimizu[†], Hidekazu Suzuki[†], Katsuhiro Naito[‡], Akira Watanabe[†]
[†]Graduate School of Science and Technology, Meijo University, Japan
[‡]Faculty of Information Science, Aichi Institute of Technology, Japan

Watanabe Lab.

1. Introduction

- Restrictions of existing networks
  - NAT traversal problem
  - Incompatibility between IPv4 and IPv6
  - Mobility

→ **NTMobile** removes all the above restrictions.

- NTMobile framework (NTMfw)
  - –NTMfw is the application library that executes NTMobile functions.
    - –NTMfw is written in C-language.

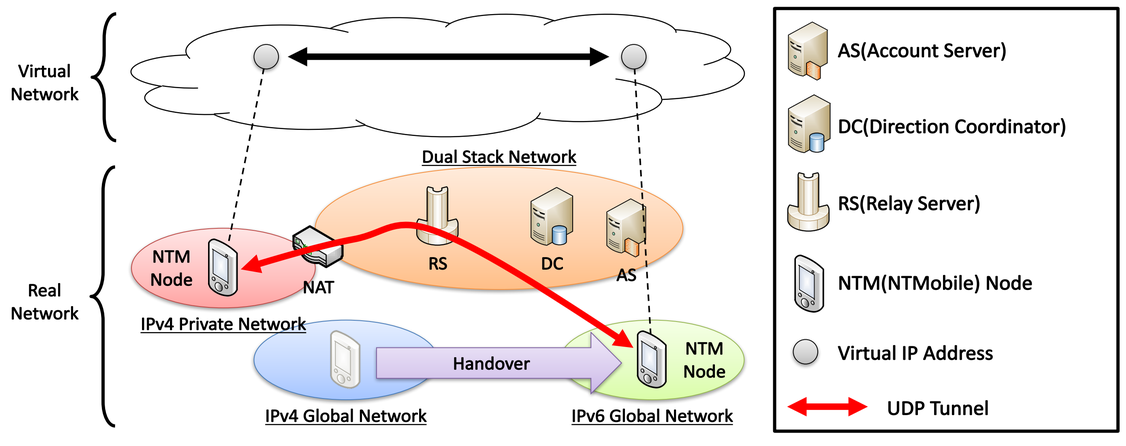
→ It can be called only by **C applications**.

- Java wrapper
  - –Java wrapper enables Java applications to use NTMfw.

→ NTMobile has become **easier** to use.

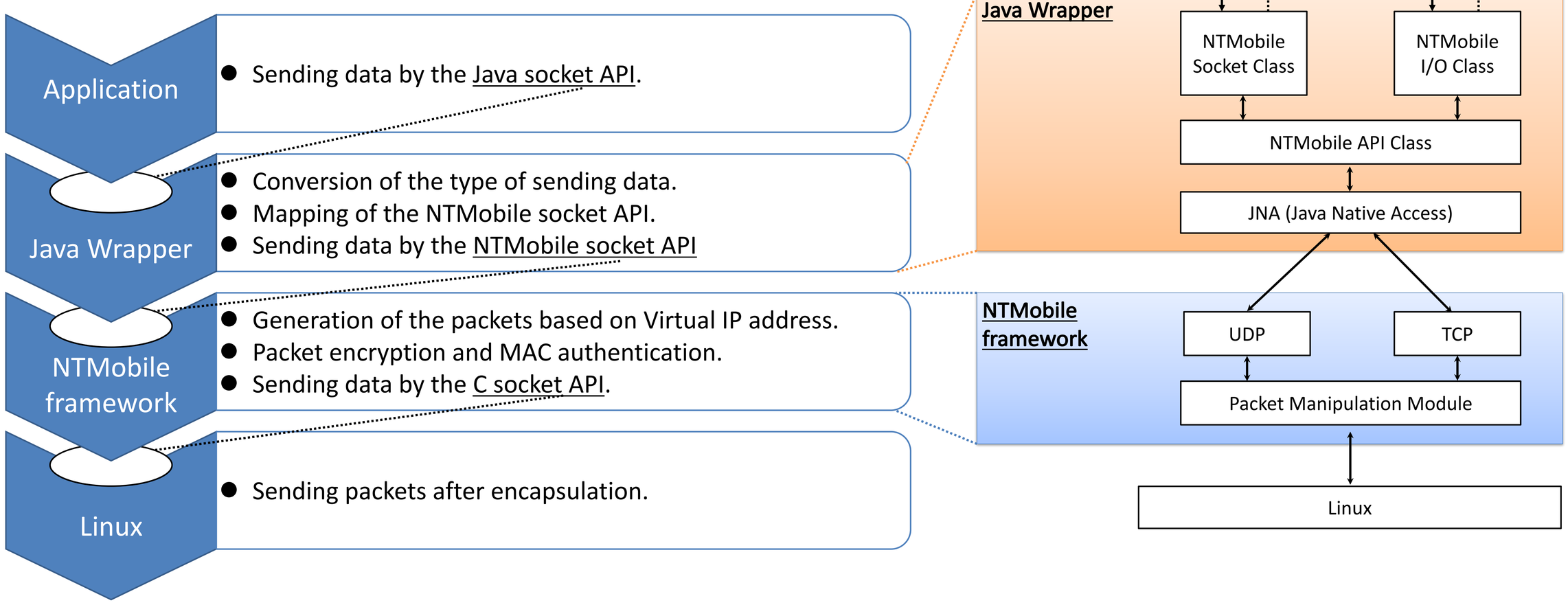
2. Network Traversal with Mobility (NTMobile)

- The technology to solve restrictions of existing networks
  - –Applications establish connections based on Virtual IP address.
  - –A packet based on Virtual IP address is encapsulated by Real IP address.
  - –NTMobile is realized in the user space by using NTMobile framework library(NTMfw).



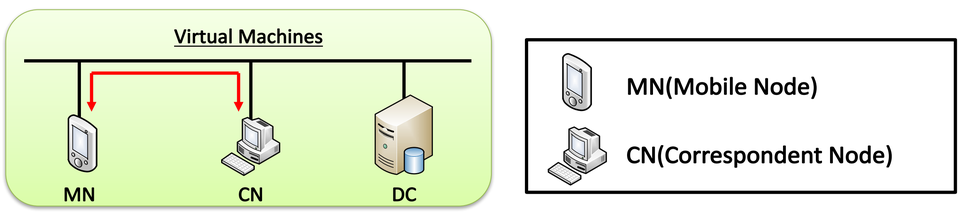
3. Structure of Java Wrapper

- Java Wrapper
  - –Creates the NTMobile socket class inheriting Java standard socket class.
  - –Defines NTMobile socket API in NTMobile socket class, and calls NTMobile framework library.



4. Evaluation

- Measurement results of the processing time.
  - –(Average of 100 times measurements)



Item	Sending time[ms]	Receiving Time [ms]
Java Wrapper	0.84	0.18
NTMfw	0.48	1.23
Total	**1.32**	**1.41**

5. Future Work

- Realization of wrappers in other languages such as Python.
- Addition of supporting communication protocols such as HTTP.