External Dynamic Mapping Method for NAT Traversal

Hidekazu Suzuki *, Yuji Goto † and Akira Watanabe ‡
Graduate School of Science and Technology, Meijo University
1-501 Shiogamaguchi, Tempaku-ku, Nagoya, 468-8502, JAPAN
Tel: +81-52-838-2406, Fax: +81-52-838-2406
E-mail: *m0641506@ccmailg.meijo-u.ac.jp, †m0732016@ccmailg.meijo-u.ac.jp
‡wtnbakr@ccmfs.meijo-u.ac.jp

Abstract— **On the Internet, it is not possible to initiate communication to internal nodes located behind a Network Address Translator (NAT) from external nodes. Therefore, we need to have a NAT traversal technology that can establish a connection between the external and internal nodes. Technologies thus far often depend on specific applications and their usage is rather limited. There are other methods which do not depend on any applications, but their efficiency for end-to-end communication is usually lowered quite a lot because they need a specific server that relays packets. This paper presents an external dynamic mapping method to solve such problems. We also define NAT-free (NAT-f) protocol to realize the method. A NAT mapping is created by a negotiation between an external node and a home gateway at the time when the external node initiates communication with an internal node. The kernel in the external node translates the address and the port number in the sending packet to a mapped-address. We have implemented and evaluated a trial system, and the results show that there is almost no performance degradation.**

I. Introduction

In recent years, peer-to-peer communications over the Internet such as IP telephony and multimedia communications have been increasing in broadband networks. However, an external node (EN) located in a global IPv4 address space cannot initiate communication to an internal node (IN) behind NAT [1] such as that in a home network. It is widely known as the NAT traversal problem. The cause of the problem is that a NAT mapping is essentially created only when an IN starts communication with an EN. Although it is possible to set the NAT mapping manually to solve the problem, it lacks flexibility because only one node can be related for each port number. IPv6 technology, that does not require NATs, has not spread to home networks yet. As a result, demand for NAT traversal technologies is increasing. The term "NAT" in this paper includes the Network Address Port Translator (NAPT) [2] that translates TCP/UDP port numbers along with the IP addresses.

Several NAT traversal technologies have been proposed hitherto. Some of them can use the home gateway already set up in home networks as it is, but they have to implement a function on the user application and are consequently lacking in flexibility. Other technologies, that do not have such problems, have other problems such as poor performance, etc. As a common problem for both technologies, a special server is always required. Otherwise, ENs can never initiate communication with INs. Then, quite a high reliability is required of the server.

Our objective is to realize a NAT traversal technology that does not depend on any application nor need a special server. To achieve the objective, we propose an external dynamic mapping method and its protocol called "NAT-free" (NAT-f). In this method, an EN negotiates with a home gateway by using NAT-f before the start of a TCP/UDP communication to create a NAT mapping. The EN directly obtains from the home gateway a the "mapped-address", namely a set of the external global IP address of the home gateway and the external port number allocated by NAT. After that, the EN translates the destination IP address and the port number contained in the packet to be sent to the IN into the mapped-address to correspond to in the NAT mapping in the kernel. In this method, the home gateway executes normal IP address/port number translation using the NAT function just as usual. As a result, our method does not cause any performance degradation in the communication.

We have implemented NAT-f in the kernel of FreeBSD and evaluated our trial system. The results of our evaluation show that the delay at the beginning of the communication is quite little, and there is almost no degradation in performance.

We explain existing technologies in Section II and present the external dynamic mapping method in Section III. We describe the implementation and the evaluation of our trial system in Section IV, and summarize the paper in Section V.

II. Existing Technologies

NAT traversal technologies can be roughly categorized into the following three types; namely, NAT behavior-based type, NAT control-based type, and NAT-less type. The NAT behavior-based type technology can use regular NATs without any modification. This type depends on the application. The NAT control-based type technology creates NAT mappings by adding functions to a home gateway. The NAT-less type technology solves the problem by its own process without the gateway performing any NAT function. Most of the latter two technologies are independent of applications.

A. NAT behavior-based type technology

Simple Traversal of UDP Through Network Address Translators (STUN) defined in RFC 3489 [3], is a protocol which utilizes a method of UDP hole punching [4], [5]. Applications in the IN need to involve STUN client functions, and send a request packet to the STUN server on the Internet. At such time, a mapped-address to the IN is created in the home gateway. The server then gets the mapped-address from the request packet received and saves it in its memory. The EN acquires the mapped-address from the server, and initiates communication to the IN by sending the mapped-address. However, STUN is not able to work with Symmetric NAT and does not support TCP-based communication, either.

Traversal Using Relay NAT (TURN) has been proposed as an additional mechanism to STUN by Internet Engineering Task Force (IETF) [6]. TURN solves the problem of STUN concerning Symmetric NAT relaying packets through the TURN server on the Internet. However, this method loses the real-time property that is needed for end-to-end communication because of the redundant routing.

Besides the above, Teredo [7] has been proposed as an IPv6 over UDP/IPv4 technology using a tunneling mechanism. Interactive Connectivity Establishment (ICE) [8] provides NAT traversal communication to Session Initiation Protocol (SIP) [9] by using the existing protocols such as STUN and TURN.

B. NAT control-based type technology

Universal Plug and Play (UPnP) [10] is a set of protocols proposed by the UPnP Forum. UPnP allows the IN to automatically create a static NAT mapping. This is essentially an automated process of port-forwarding function. This technology is already implemented in many NATs and widely used. However, an IN has to report the mapped-address to the server on the Internet as in the case of STUN, when an EN initiates communication with an IN. NAT Port Mapping Protocol [11] has been also proposed for the same purpose.

Address Virtualization Enabling Service (AVES) is a network-layer waypoint service proposed in [12]. In AVES, ENs and INs can be used without implementing any special functions, however, it needs special devices such as an AVES-aware DNS server and a waypoint which encapsulates and relays communication packets. The NAT also needs to be modified so that it can handle AVES protocols and encapsulated packets. That means that there is certain performance degradation.

C. NAT-less type technology

"4+4" is an address extension architecture proposed in [13]. It extends an IPv4 header to contain two types of addresses; namely, a global IP address of a home gateway and a private IP address of an IN. The gateway forwards packets sent from the EN to the IN by swapping the addresses. That is to say that the gateway does not undertake any NAT mapping, but instead executes its own routing process. This method is, however, not practical because all of the ENs, INs and home gateways should modify the protocol stack. Also, it may affect other systems because it changes the packet format.

IP Next Layer (IPNL) [14] has also been proposed as a technology which uses its own routing process.

III. Our Proposed Method

We propose an external dynamic mapping method as a new NAT traversal technology. Although it is categorized as one type of the NAT control-based type technology, it does not need any special server such as a STUN/TURN server. We also define a protocol called "NAT-free" (NAT-f) to realize the method. In our proposed method, an EN instructs a home gateway to create a NAT mapping and obtains the mapped-address by NAT-f before the start of communication with an IN. Then, the EN translates the IP address and the port number in a packet into the mapped-address so as to correspond to in the NAT mapping.

A. Overview

Fig. 1 shows the system configuration and the initial setting of our proposed method. NAT-f is implemented both in the EN and in the home gateway. The Dynamic DNS (DDNS) [15] server, which is widely deployed, can be used for the name resolution of INs. This server only has to meet the requirement that a wildcard function [16] can be used. A user has to register himself in a DDNS service provider, and acquire a domain name (e.g., $home.example.net$) in advance. The DDNS server saves the association between the domain name of the home gateway and IP address $G2$ as a wildcard A record. The home gateway has an Access Control Table (ACT) which includes the following information (an IN's name, private IP address, and access control flag):

$$alice := (P1, allow), \qquad bob := (P2, deny)$$

The name of the IN can be determined freely, as long as the same is kept in the home network. We call this name "private host name" (PHN) to distinguish it from a general host name registered in the DDNS server (e.g., $home$). The access control flag indicates whether an EN is allowed to access to the corresponding IN or not.

Fig. 2 shows the communication sequence at the time when an EN starts communication with an IN ($alice$) by our

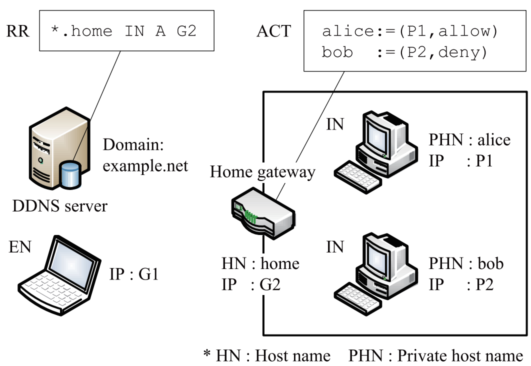


Fig. 1. System configuration and initial setting.

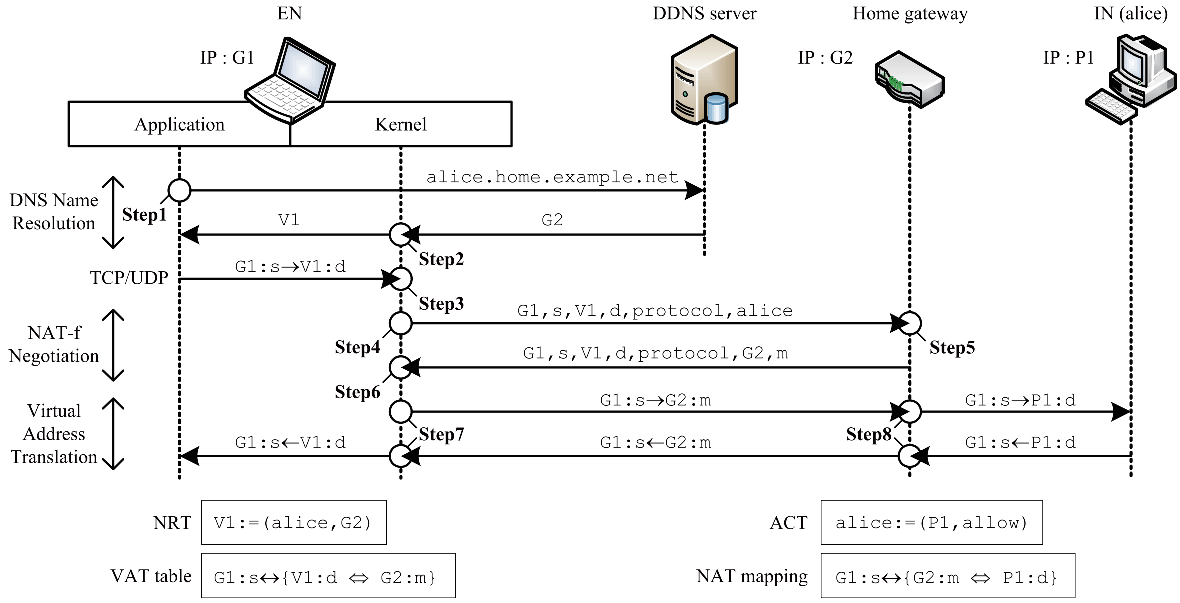


Fig. 2. Communication sequence with our proposed method.

proposed method. Here, TCP/UDP packets are described as follows:

$$A : a \rightarrow B : b,$$

where A and a are the source IP address and the port number, and B and b are the destination IP address and the port number. Our proposed method is composed of three phases as described below:

1) DNS Name Resolution:

Step1: The EN sends a query to the DDNS server with the name that is composed of the PHN of the IN and FQDN (i.e., $alice.home.example.net$) or the home gateway. The DDNS server replies to the IP address of the home gateway ($G2$) through the wildcard function.

Step2: The EN hooks the DNS reply packet in the kernel and changes the IP address ($G2$) into a virtual IP address ($V1$). This address is associated with the PHN and the IP address of the home gateway as follows:

$$V1 := (alice, G2)$$

This information is cached in the Name Relation Table (NRT) of the EN. After that, the virtual IP address is reported to the upper layer, and the application recognizes that the IP address of the IN ($alice$) is $V1$.

2) NAT-f Negotiation:

Step3: The application creates TCP/UDP packets in which the destination IP address is made the virtual IP address, and passes it to the kernel. The kernel refers to its Virtual Address Translation (VAT) table, which has entries about the relationship between the virtual IP address and the mapped-address in the home gateway, based on the source and destination IP addresses/port numbers, and the protocol type. The VAT entry is created when a NAT-f negotiation is completed (Step 6). If the EN already has the corresponding VAT entry, the EN jumps to Step 7.

Step4: If no VAT entry exists, the EN searches the NRT by referring to the destination IP address (i.e., the virtual IP address, $V1$) and obtains associated information. After that, the TCP/UDP packet is temporarily stored in the kernel memory and the EN starts a NAT-f negotiation. A NAT-f mapping request, which contains the source and destination IP addresses/port numbers, the protocol type, and the PHN, is sent to the home gateway.

Step5: The home gateway obtains the information from the NAT-f mapping request and checks the ACT. If the ACT has a corresponding PHN and its access control flag shows "$allow$", the home gateway creates a NAT mapping based on the information in the packet received and the ACT. When the mapped-address is $G2 : m$, we get the NAT mapping as follows:

$$G1 : s \leftrightarrow \{G2 : m \overset{NAT}{\Longleftrightarrow} P1 : d\}$$

Here, "$\overset{NAT}{\Longleftrightarrow}$" means translation between both sides by the NAT, and "$G1 : s \leftrightarrow P1 : d$", which is derived from the above description, means communication between the EN and the IN ($alice$). The home gateway makes a reply packet that contains the information received from the EN and the mapped-address, and sends it to the EN.

Step6: The EN creates the VAT entry when it receives the above reply as follows:

$$G1 : s \leftrightarrow \{V1 : d \overset{VAT}{\Longleftrightarrow} G2 : m\}$$

Next, the EN restores the stored TCP/UDP packet. In this way, the NAT-f negotiation is completed.

3) Virtual Address Translation:

Step7: As for the restored TCP/UDP packet, the destination IP address and the port number are translated from $V1 : d$ to $G2 : m$ according to the VAT table, and the packet is sent to the home gateway.

Step8: The home gateway handles the received packet in the normal NAT method, and the destination IP address and the port number of the packet are translated from $G2 : m$ to $P1 : d$. Then, the packet is forwarded to the corresponding IN ($alice$).

As for the reply packet from the IN to the EN, the reserve translation as stated above is performed.

B. Characteristics of NAT-f

1) Support for TCP communication: The protocol type of the packet to be sent is notified in the NAT-f negotiation in the way as described in Step 3. As a result, NAT-f can deal with both TCP and UDP because NAT mappings are created corresponding to the protocol type.

2) Support for Symmetric NAT: The NAT-f negotiation is executed each time when the source or the destination port number contained in the packet sent by an EN is changed. NAT mappings are created in the home gateway for every connection established between an EN and an IN. Consequently, NAT-f supports both types of the Cone NAT and the Symmetric NAT.

3) Simultaneous communication for INs: An EN can communicate with multiple INs in a home network simultaneously by using the same destination port numbers. Now, we assume a case where an EN is about to start communication with bob while it is communicating with $alice$ in Fig. 1. The communication between the EN and $alice$ can be shown as follows:

$$G1 : s \leftrightarrow \{V1 : d \overset{VAT}{\Longleftrightarrow} G2 : m \overset{NAT}{\Longleftrightarrow} P1 : d\}$$

The EN changes the IP address of the home gateway into a virtual IP address ($V2$) after receiving a DNS query reply about bob. An application recognizes the destination IP address as $V1$ for $alice$, and $V2$ for bob. Therefore, a NAT mapping corresponding to bob is newly created as $G2 : n$ in the home gateway. Consequently, the communication between the EN and bob can be described as follows:

$$G1 : t \leftrightarrow \{V2 : d \overset{VAT}{\Longleftrightarrow} G2 : n \overset{NAT}{\Longleftrightarrow} P2 : d\},$$

where t is a source port number dynamically allocated by the kernel.

4) Communication for non-NAT-f-compliant devices: NAT-f mapping request/reply is based on ICMP Echo. If a target device (i.e., the home gateway or IN [1]) does not support NAT-f protocol, it replies ICMP Echo Reply having the same content in ICMP Echo. The EN can judge whether the target supports NAT-f or not from the reply packet. In this case, the EN creates the VAT entry to return the original IP address of the target from the virtual IP address as follows:

$$G1 : s \leftrightarrow \{V1 : d \overset{VAT}{\Longleftrightarrow} G2 : d\}$$

After that, the EN starts communication with the target.

5) Private-to-Private communication: Our proposed method can be easily extended to the communication between INs located in different home networks. In this case, the functions implemented in the EN (i.e., DNS rewriting, NAT-f negotiation, and virtual address translation process) have to be implemented in the home gateways. The NAT-f negotiation and the virtual address translation are executed on both sides of the home gateways.

[1]In case that the IN locates on the Internet, not behind a NAT.

IV. Evaluations

A. Implementation

We have implemented the NAT-f module in the IP layer of FreeBSD 6.1-RELEASE and have made a prototype system. Fig. 3 and Fig. 4 show the implementation of NAT-f module in an EN and in a home gateway. The module consists of the three functions; namely, DNS rewriting, negotiation, and VAT functions, and it is called from the input/output functions — `ip_input()` and `ip_output()` — in the IP layer. Packets are processed by the NAT-f module and returned to the original place.

The negotiation function handles NAT-f mapping request/reply packets and creates VAT entries in the EN and NAT mappings in the home gateway. The VAT function handles TCP/UDP packets to translate the virtual IP address from/to the mapped-address based on VAT entries. If no VAT entry exists, the TCP/UDP packet is temporarily stored in the kernel memory and the VAT function calls the negotiation function to start a NAT-f negotiation. The packet is restored immediately and passed to `ip_output()` just after the NAT-f negotiation has completed. The VAT table and the NRT are implemented as a hash table. The life time of information cached in the NRT is made the TTL value described in a DNS reply packet.

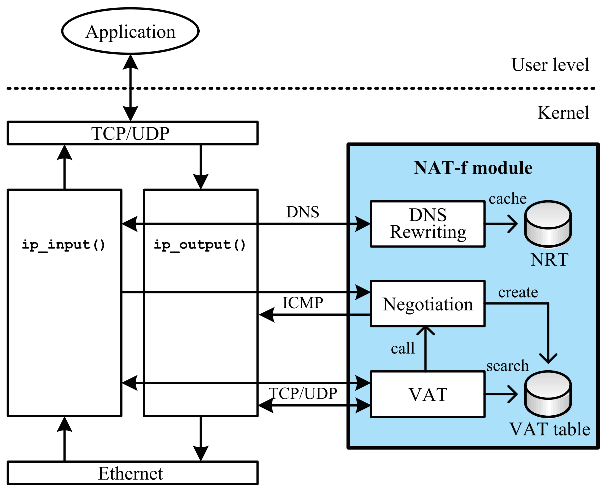


Fig. 3. Implementation of NAT-f in EN.

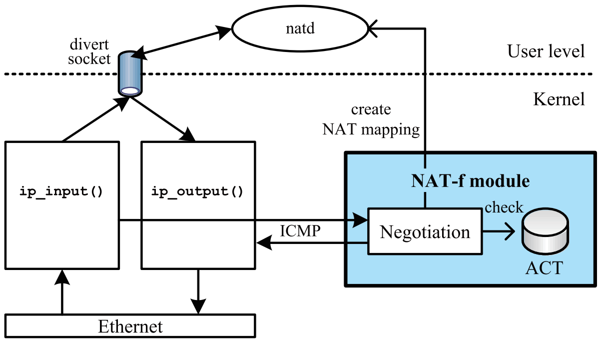


Fig. 4. Implementation of NAT-f in home gateway.

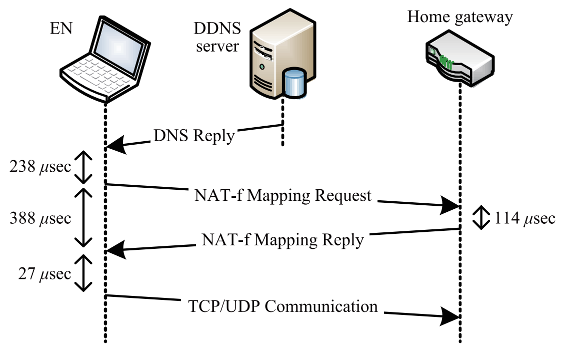


Fig. 5. Initial delay caused by NAT-f.

TABLE I

INTERNAL PROCESS TIME OF THE NAT-F MODULE

	EN	Home gateway
Step 2	7.1 μsec	—
Step 3	0.8 μsec	—
Step 4	2.9 μsec	—
Step 5	—	114.2 μsec
Step 6	1.4 μsec	—
Step 7	1.1 μsec	—

TABLE II

THROUGHPUT ON OUR PROPOSED METHOD

Message size	TCP Stream (Mbps)		UDP Stream (Mbps)	
(Bytes)	EN → IN [a]	EN ← IN [b]	EN → IN	EN ← IN
64	93.2	93.1	49.3	49.3
128	93.2	93.2	66.0	66.0
256	93.2	93.2	79.6	79.6
512	93.2	93.2	88.8	88.8
1024	93.2	93.2	94.4	96.4
1472	93.2	93.2	96.4	96.4

[a] In case of the NAT traversal communication on our proposed method.
[b] In case of the ordinary communication. (i.e., NAT-f is not implemented.)

The VAT entry is deleted with the kernel timer after a certain period of time under the idle state or when TCP connections are disconnected.

A virtual IP address is allocated corresponding to the private host name of the IN. When a virtual IP address is described as "$W.X.Y.Z$", the following values are established to individual portions. "W" is allocated as the IP address of class E, and the NAT-f module identifies whether it is a virtual IP address or not based on this value. "X" is made 0 as the initial value. "Y" is made the hash value of the private host name of the IN, and "Z" is made the hash value of the domain name of the IN. The range of the hash value is between 1 to 254. When the hash value collides, "X" is given a different value to prevent duplication.

B. Performance

We measured the performance of our proposed method in the system configuration shown in Fig. 1. The specification for each device is based on Pentium 4 3.0 GHz of CPU and 512 MB of memory. The EN, the home gateway, and the DDNS server are connected by a switch device. We used a network analyzer, Ethereal to measure the initial delay caused by the NAT-f protocol, and Read Time Stamp Counter (RDTSC) [17] to measure the internal process time of the NAT-f module. We also measured the TCP/UDP throughput between the EN and the IN by using Netperf in order to clarify the influence of the virtual address translation process executed in an EN on the communication performance. For the sake of comparison, we also measured the throughput in the case where NAT-f is not implemented in the system.

1) Initial delay caused by NAT-f: Fig. 5 shows the results of the measured time, and Table I shows the internal process time of the NAT-f module. It took 238 μsec from the reception of the DNS reply packet to the transmission of the NAT-f mapping request packet in the EN. In This process, 7.1 μsec was for the changing process of the DNS reply packet (Step 2) and 3.4 μsec was for the starting process of the NAT-f negotiation (Step 3 and Step 4). It took 388 μsec from the transmission of the mapping request to the reception of the mapping reply in the EN (Step 5 was 114.2 μsec out of this.) It also took 27 μsec from the reception of the mapping reply to the transmission of the restored TCP/UDP packet in the EN. In fact, the initial delay caused by NAT-f was only about 650 μsec [2]. This means that our proposed method scarcely affects the TCP/UDP communication. This is because of the reason that the EN stored and restored the TCP/UDP packet, which triggers the NAT-f negotiation, in the kernel memory. With this method, TCP retransmission never occurs at the time when communication starts.

2) TCP/UDP throughput: The result of the throughput between the EN and the IN is shown in Table II. A throughput comparison shows no significant difference between the case that NAT-f is implemented and the case that NAT-f is not implemented for both TCP and UDP and in any message size. From the above result, we can conclude that the process of virtual address translation executed in the EN rarely affects the TCP/UDP throughput.

[2] 238+388+27=650 μsec

TABLE III
COMPARISON WITH EXISTING TECHNOLOGIES

<table>
<tr><th></th><th>STUN</th><th>TURN</th><th>ICE</th><th>UPnP</th><th>AVES</th><th>NAT-f</th><th>4+4</th></tr>
<tr><td>Type of NAT traversal technology</td><td colspan="3">NAT behavior-based</td><td colspan="3">NAT control-based</td><td>NAT-less</td></tr>
<tr><td>Modification to a home gateway</td><td colspan="3">Not required</td><td colspan="2">Application</td><td colspan="2">Kernel</td></tr>
<tr><td>Installation of functions on EN</td><td colspan="4">Application</td><td>Not required</td><td colspan="2">Kernel</td></tr>
<tr><td>Installation of functions on IN</td><td colspan="4">Application</td><td colspan="2">Not required</td><td>Kernel</td></tr>
<tr><td>Required special servers</td><td>STUN server</td><td>TURN server</td><td>STUN server
TURN server</td><td>Application server [a]</td><td>Waypoint
DNS server [b]</td><td colspan="2">Not required</td></tr>
<tr><td>Support for TCP communication</td><td>×</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td></tr>
<tr><td>Support for Symmetric NAT</td><td>×</td><td>○</td><td>○</td><td>×</td><td>— [c]</td><td>○</td><td>— [d]</td></tr>
<tr><td>Optimal communication route</td><td>○</td><td>×</td><td>△</td><td>○</td><td>×</td><td>○</td><td>○</td></tr>
<tr><td>Throughput</td><td>○</td><td>○</td><td>○</td><td>○</td><td>×</td><td>○</td><td>○</td></tr>
</table>

[a] An application server is required so that an EN can acquire the NAT mapping created by an IN with the UPnP protocol.
[b] This server is called "AVES-aware DNS server" and is required the specification that specializes in this technology.
[c] The normal NAT mapping function is not undertaken. The NAT executes its own decapsulation process and address translation process.
[d] The normal NAT mapping function is not undertaken. The NAT executes its own routing process.

C. *Comparison with existing technologies*

A comparison between our proposed method and the existing technologies is shown in Table III. Our proposed method "NAT-f" has several advantages over the existing technologies, although the home gateway needs to be modified. In NAT-f, NAT mappings can be created by the action from the outside of the home gateway, or an EN. This means that home information appliances which will become increasingly popular do not need any special functions to realize NAT traversal. (In the case of STUN, TURN, ICE, and UPnP, NAT mappings have to be created by the action from the inside of the home gateway, or an IN.)

Our proposed method can also support TCP communication and Symmetric NAT as mentioned above. Moreover, we need no special servers because the EN obtains mapped-addresses from the home gateway directly. There is no redundant communication route, and the encapsulation process is not needed. In TURN, AVES, and possibly even ICE, all communication between EN and IN is delayed because of the redundant route caused by relaying the TURN server or waypoint. Additionally the waypoint translates and encapsulates the packets. Due to these processes, the end-to-end throughput is sure to decrease compared with other technologies. Although the EN executes virtual address translation in each packet in our system, the process rarely affects the communication performance.

V. CONCLUSION

In this paper, we have proposed the external dynamic mapping method to realize NAT traversal communication. The EN negotiates with a home gateway by using NAT-f to create NAT mappings. After completing the negotiation, the EN translates the destination IP address and the port number contained in the packet to be sent to the IN into the mapped-address. It does not depend on any application, and requires no special servers. As the result of our evaluation, it rarely affects the performance of TCP/UDP communications.

REFERENCES

[1] K. Egevang and P. Francis, "The ip network address translator (nat)," RFC 1631, May 1994.
[2] P. Srisuresh and M. Holdrege, "Ip network address translator (nat) terminology and considerations," RFC 2663, Aug. 1999.
[3] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, "Stun - simple traversal of user datagram protocol (udp) through network address translators (nats)," RFC 3489, Mar. 2003.
[4] M. Holdrege and P. Srisuresh, "Protocol complications with the ip network address translator," RFC 3027, Jan. 2001.
[5] B. Ford, P. Srisuresh, and D. Kegel, "Peer-to-peer communication across network address translators," in *Proc. USENIX Annual Technical Conference*, Anaheim, CA, Apr. 2005, pp. 179–192.
[6] J. Rosenberg, R. Mahy, and C. Huitema. (2005) Traversal using relay nat (turn). Internet draft. [Online]. Available: http://tools.ietf.org/id/draft-rosenberg-midcom-turn-08.txt
[7] C. Huitema, "Teredo: Tunneling ipv6 over udp through network address translations (nats)," RFC 4380, Feb. 2006.
[8] J. Rosenberg. (2006) Interactive connectivity establishment (ice): A methodology for network address translator (nat) traversal for offer/answer protocols. Internet draft. [Online]. Available: http://tools.ietf.org/id/draft-ietf-mmusic-ice-12.txt
[9] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "Sip: Session initiation protocol," RFC 3261, June 2002.
[10] (2001, Nov.) Internet gateway device (igd) standardized device control protocol v 1.0. UPnP Forum. [Online]. Available: http://www.upnp.org/standardizeddcps/documents/UPnP_IGD_1.0.zip
[11] S. Cheshire, M. Krochmal, and K. Sekar. (2006) Nat port mapping protocol (nat-pmp). Internet draft. [Online]. Available: http://tools.ietf.org/id/draft-cheshire-nat-pmp-02.txt
[12] T. S. E. Ng, I. Stoica, and H. Zhang, "A waypoint service approach to connect heterogeneous internet address spaces," in *Proc. USENIX Annual Technical Conference*, Boston, MA, June 2001, pp. 319–332.
[13] Z. Turányi, A. Valkó, and A. Campbell, "4+4: An architecture for evolving the internet address space back toward transparency," in *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 5, New York, NY, Oct. 2003, pp. 43–54.
[14] P. Francis and R. Gummadi, "Ipnl: A natextended internet architecture," in *ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2001, pp. 69–80.
[15] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound, "Dynamic updates in the domain name system (dns update)," RFC 2136, Apr. 1997.
[16] E. Lewis, "The role of wildcards in the domain name system," RFC 4592, July 2006.
[17] (1998) Using the rdtsc instruction for performance monitoring. Intel Corp. [Online]. Available: http://developer.intel.com/drg/pentiumII/appnotes/RDTSCPM1.htm

IEEE 7th International Symposium on Communications and Information Technologies (ISCIT2007)
Oct. 16th – 19th, 2007
Crowne Plaza Hotel, Darling Harbour, Sydney, Australia

Session: T5B – Internet II

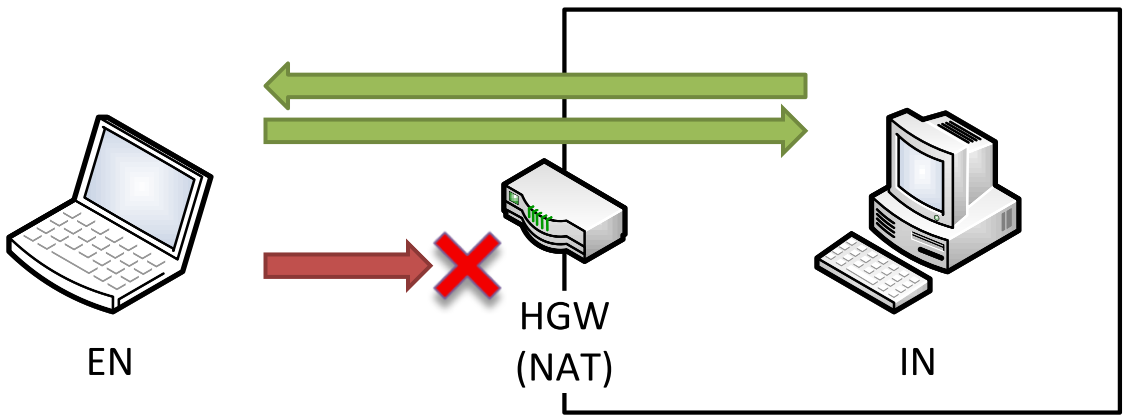
External Dynamic Mapping Method for NAT Traversal

Hidekazu Suzuki Yuji Goto Akira Watanabe
Meijo University, JAPAN

Motivation

NAT: Network Address Translator

- NAT traversal problem
  - A home network is constructed behind NAT (Home gateway)
  - It is not possible to initiate communication from External Node (EN) to Internal Node (IN)



A demand for NAT traversal technologies is increasing in End-to-End communication

Existing Major Technologies

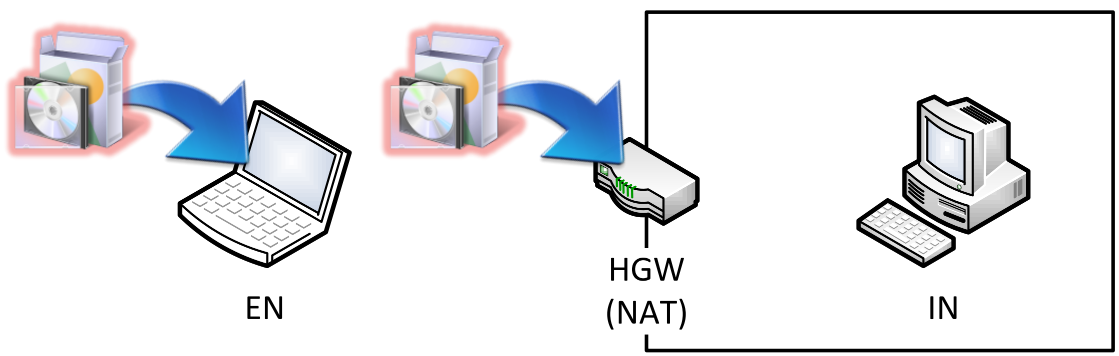
- **STUN (Simple Traversal of UDP through NAT)**
  - It utilizes a method of UDP hole punching
- **TURN (Traversal Using Relay NAT)**
  - A special server relays all packets between IN and EN
- **UPnP (Universal Plug and Play)**
  - IN automatically creates a static NAT mapping

List of issues to be solved:
✓Symmetric NAT and TCP are not supported
✓Communication delay is large
✓A Special server is indispensable
✓Applications need to have the functions

Our Proposed Method

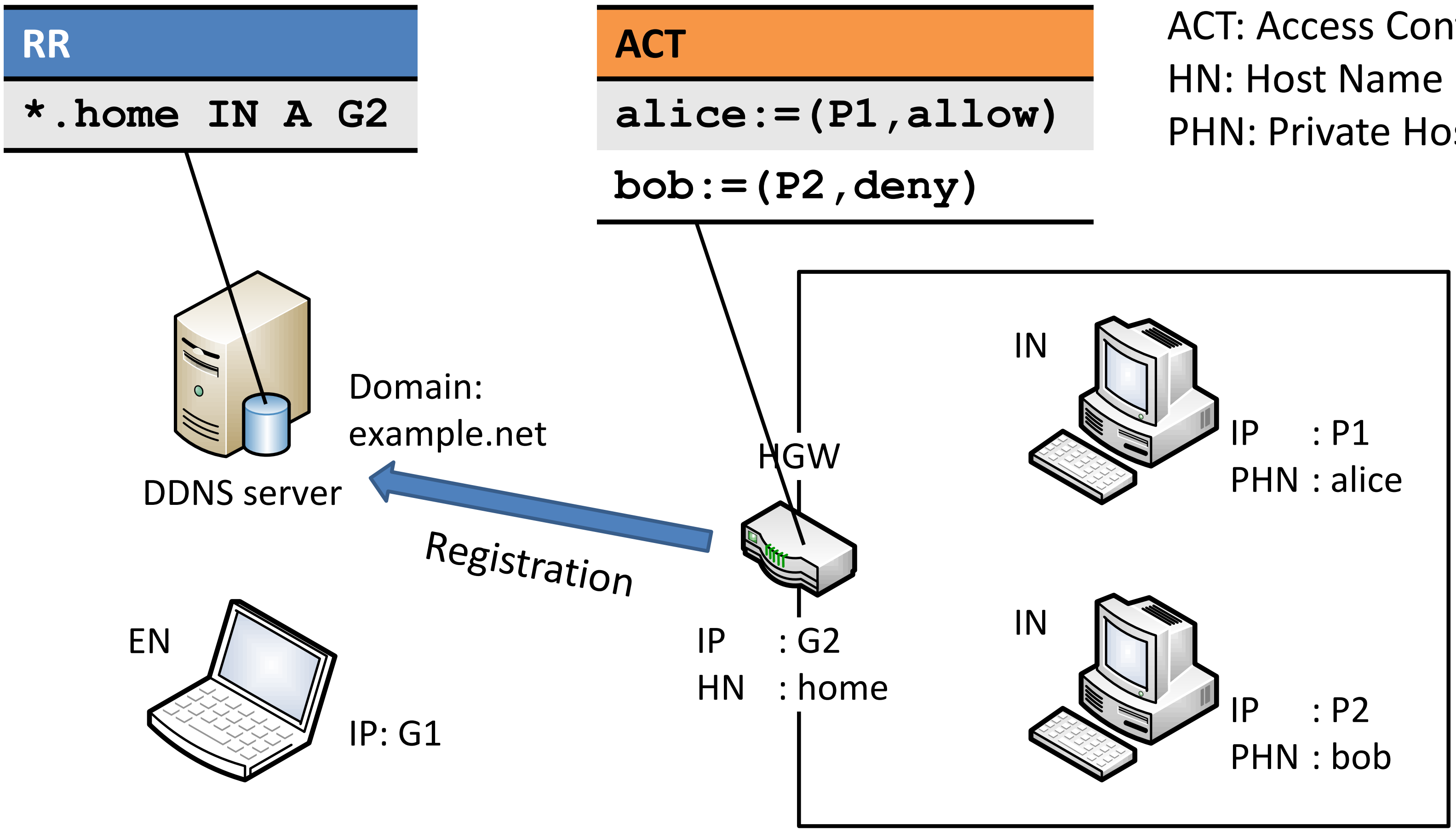
Watanabe Lab.
Department of Information Engineering, Meijo University
Information Networks Laboratory

- **External Dynamic Mapping Method**
  - It does not depend on any applications, and does not require a special server
  - EN instructs HGW to create a NAT mapping by **NAT-f protocol**
- **NAT-free protocol (NAT-f)**
  - It is implemented in the kernel of EN and HGW
  - It creates a NAT mapping in HGW and a **Virtual Address Translation (VAT) table** in EN



System Configuration and Initial Settings

DDNS: Dynamic DNS
RR: Resource Record
ACT: Access Control Table
HN: Host Name
PHN: Private Host Name



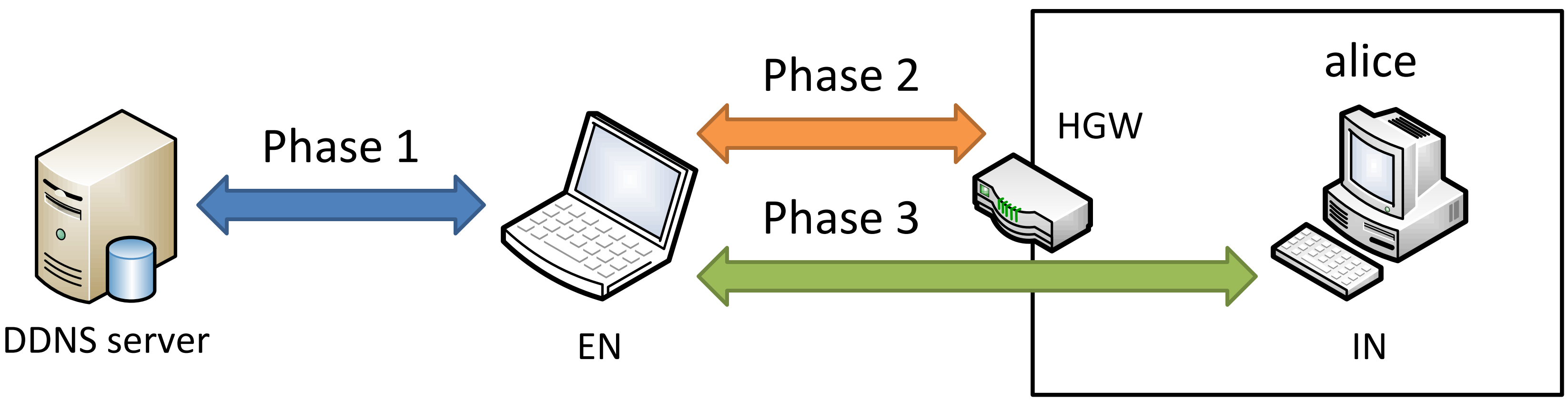
Overview of Our Proposed Method

- It is composed of three phases:

  Phase 1) DNS Name Resolution

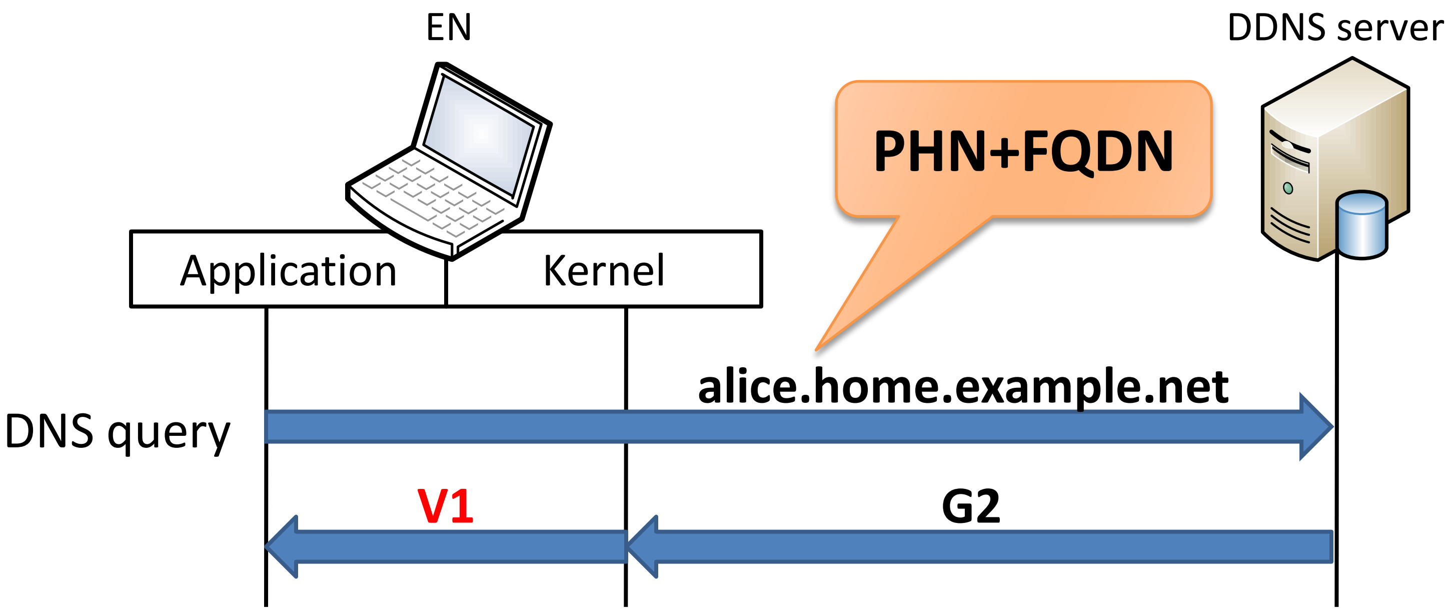
  Phase 2) NAT-f Negotiation

  Phase 3) End-to-End Communication Based on Virtual Address Translation



Ph.1) DNS Name Resolution

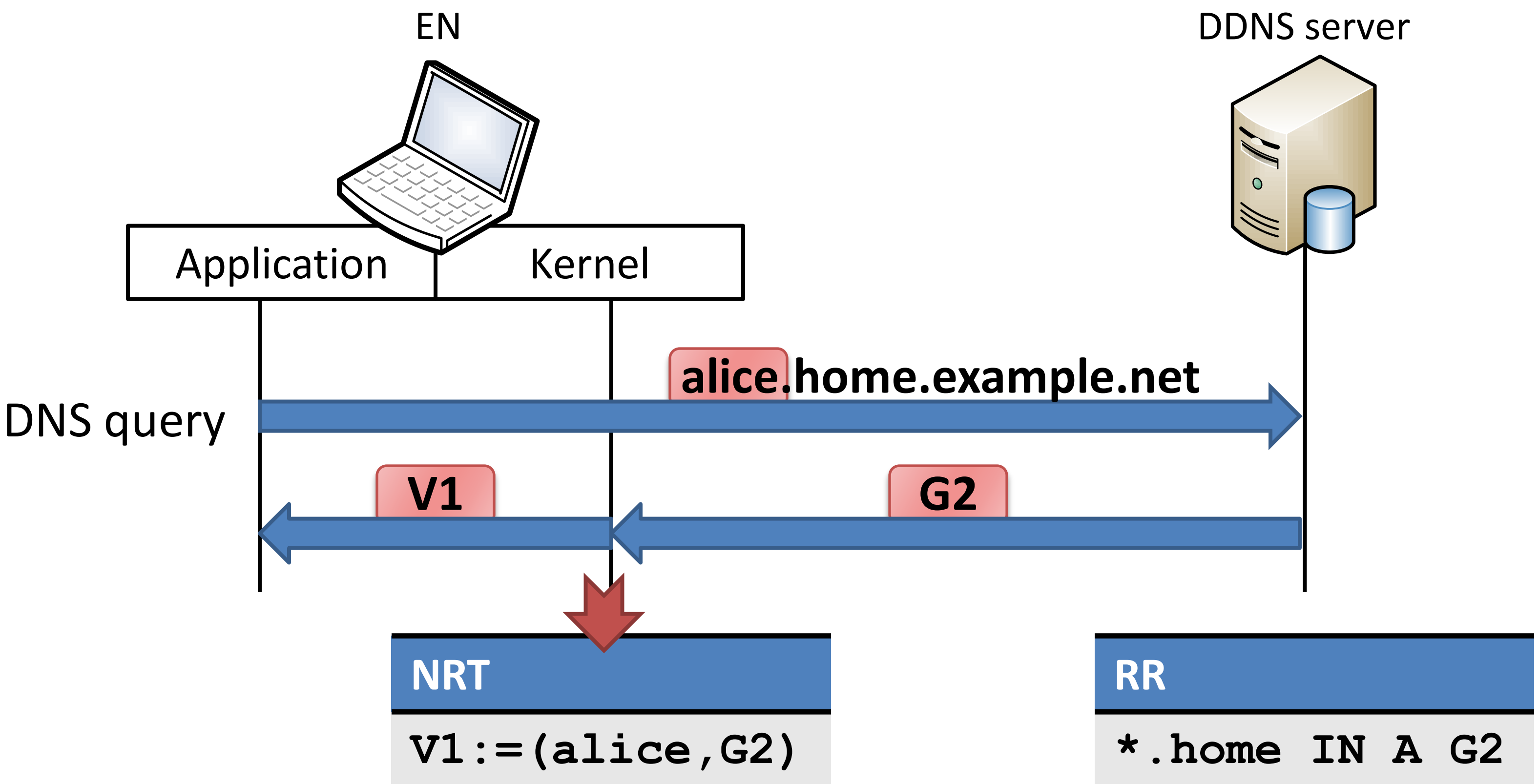
- EN resolves the domain name of alice
- EN hooks a DNS reply in the kernel and rewrites the contents
    - Obtained IP address (G2) ➔ **Virtual IP address (V1)**



RR
*.home IN A G2

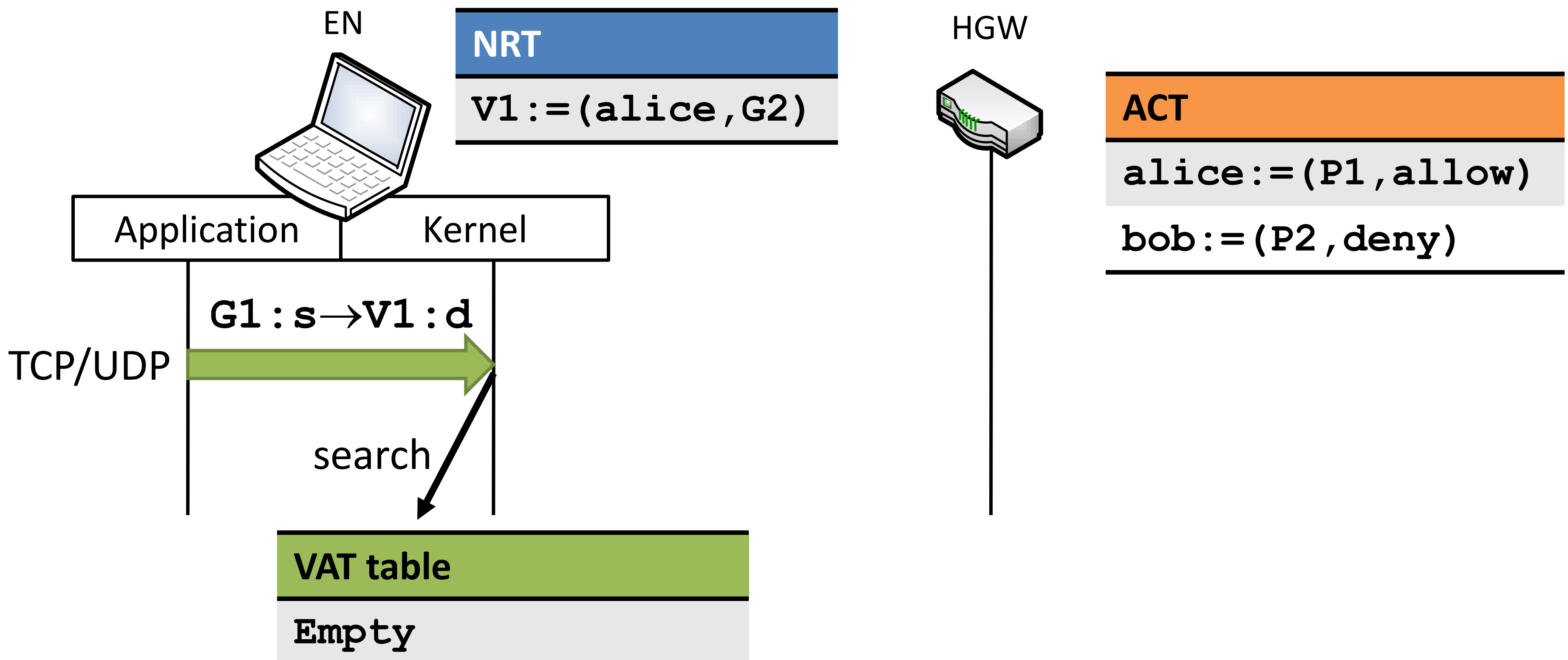
Ph.1) DNS Name Resolution

- <u>The information</u> is cached in **Name Relation Table (NRT)**
  - Virtual IP address, PHN and obtained IP address (HGW)
- An application recognizes that the IP address of alice is V1



Ph.2) NAT-f Negotiation

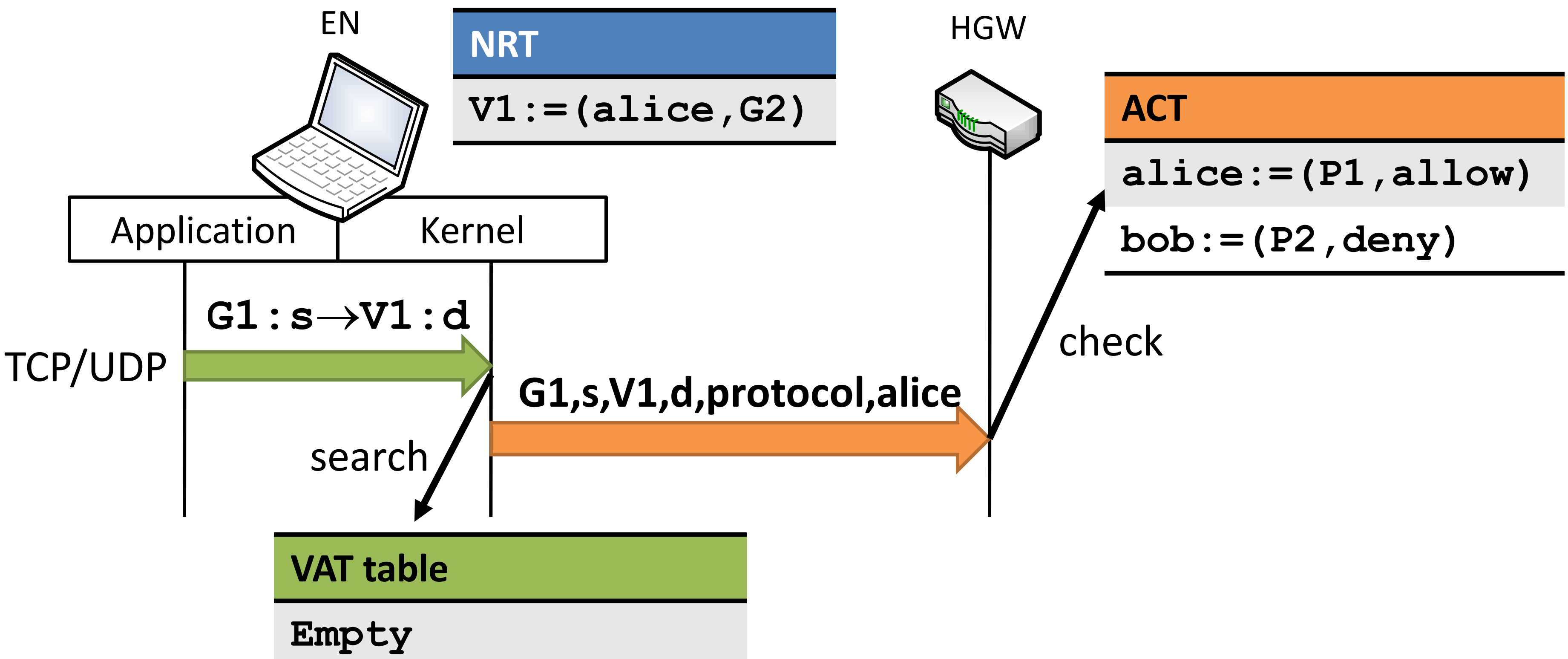
- EN searches a VAT table
  - Relationship between a virtual IP address and a mapped-address
  - No VAT entry exists yet ➔ EN temporarily stores the packet



Ph.2) NAT-f Negotiation

Watanabe Lab.
Department of Information Engineering, Meijo University
Information Networks Laboratory

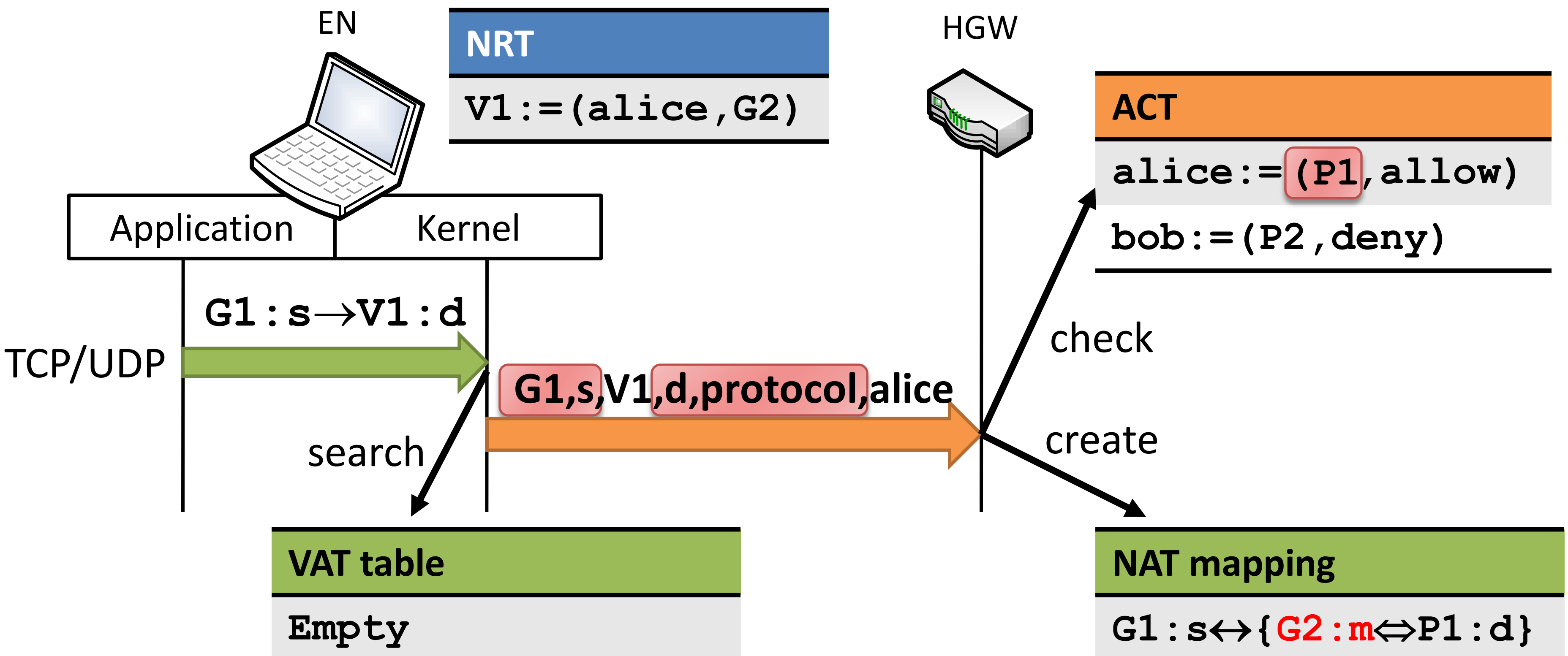
- EN sends a NAT-f mapping request to HGW
- HGW checks the ACT with the received PHN
  - If the access control flag is “allow”, …



Ph.2) NAT-f Negotiation

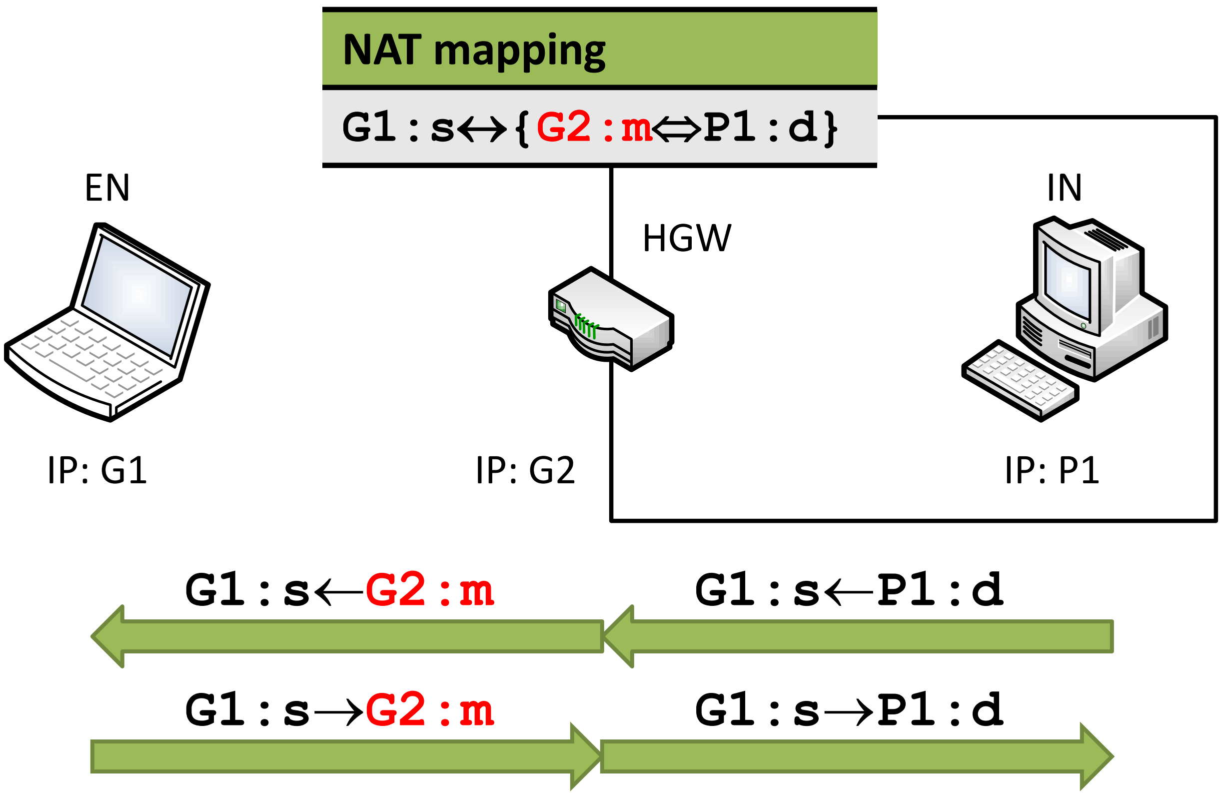
Watanabe Lab.
Department of Information Engineering, Meijo University
Information Networks Laboratory

- HGW creates a NAT mapping with
  - Received information
  - Private IP address of alice



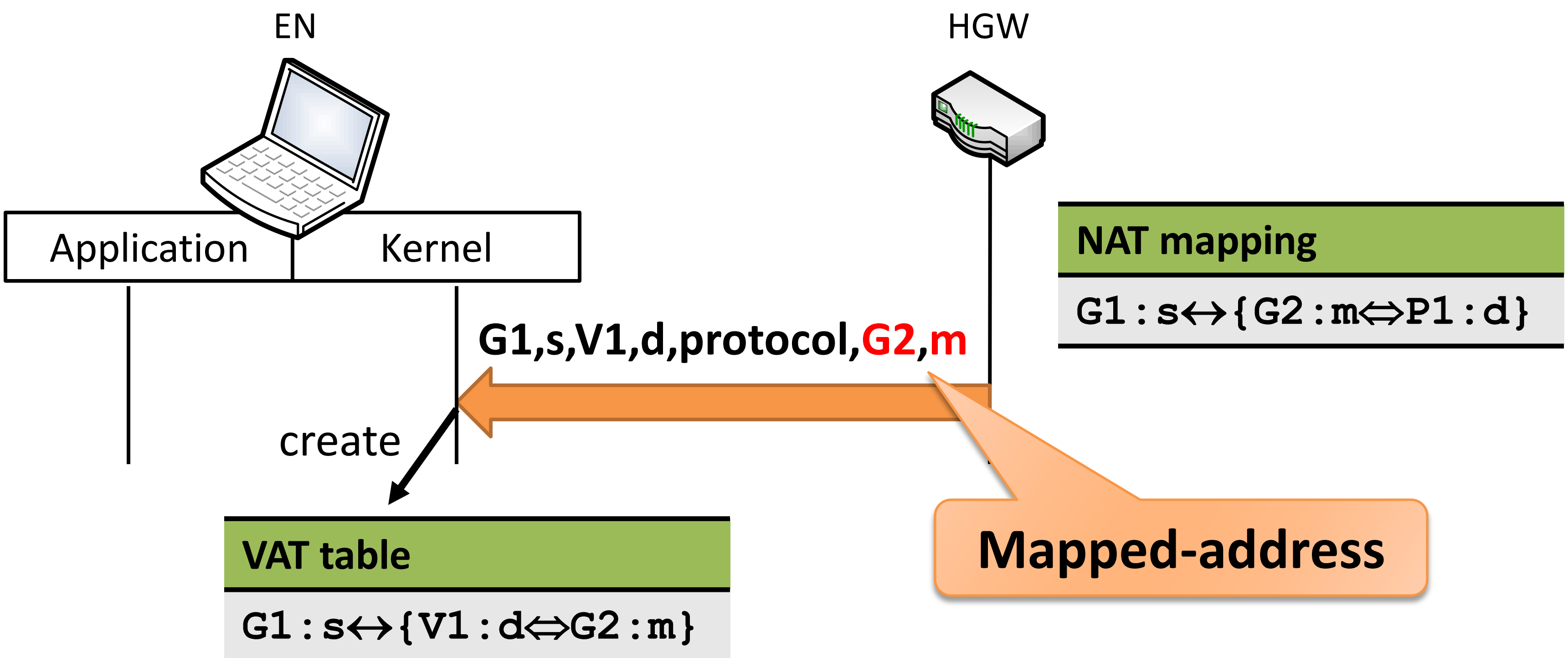
Ph.2) NAT-f Negotiation

- Mapped-address is allocated for the communication between IN and EN by the NAT function



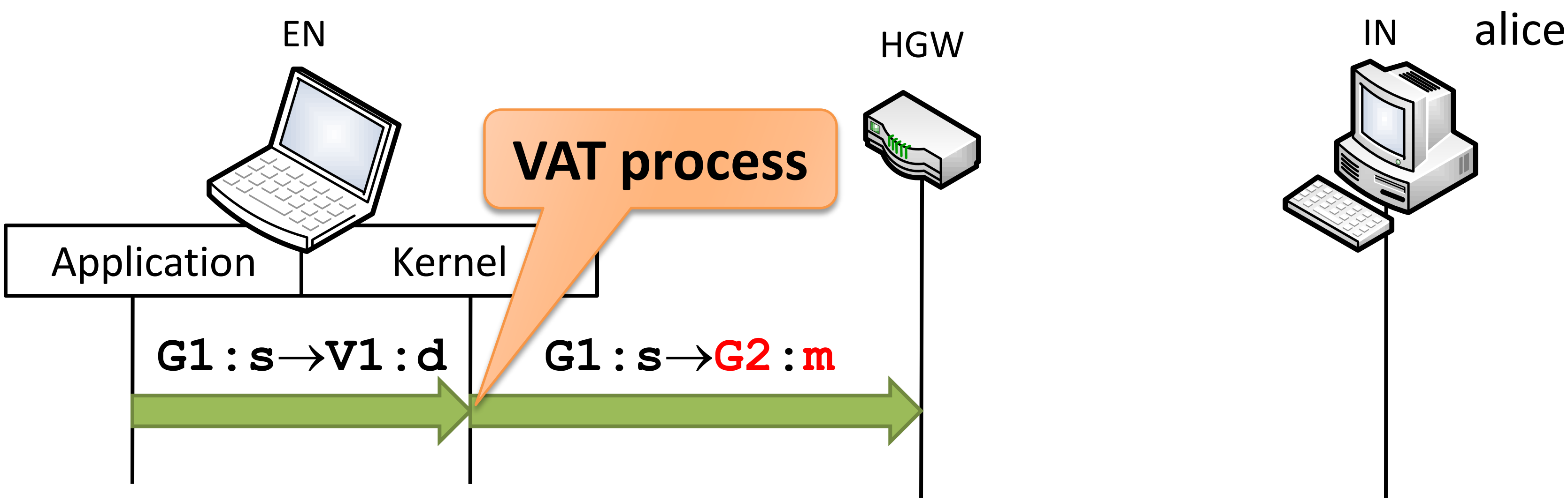
Ph.2) NAT-f Negotiation

- HGW sends a NAT-f mapping reply to notify EN of the mapped-address
- EN creates the VAT entry



Ph.3) End-to-End Communication Based on Virtual Address Translation

- EN restores the stored TCP/UDP packet and translates its destination according to the VAT table
  - Destination: V1:d ⇒ G2:m

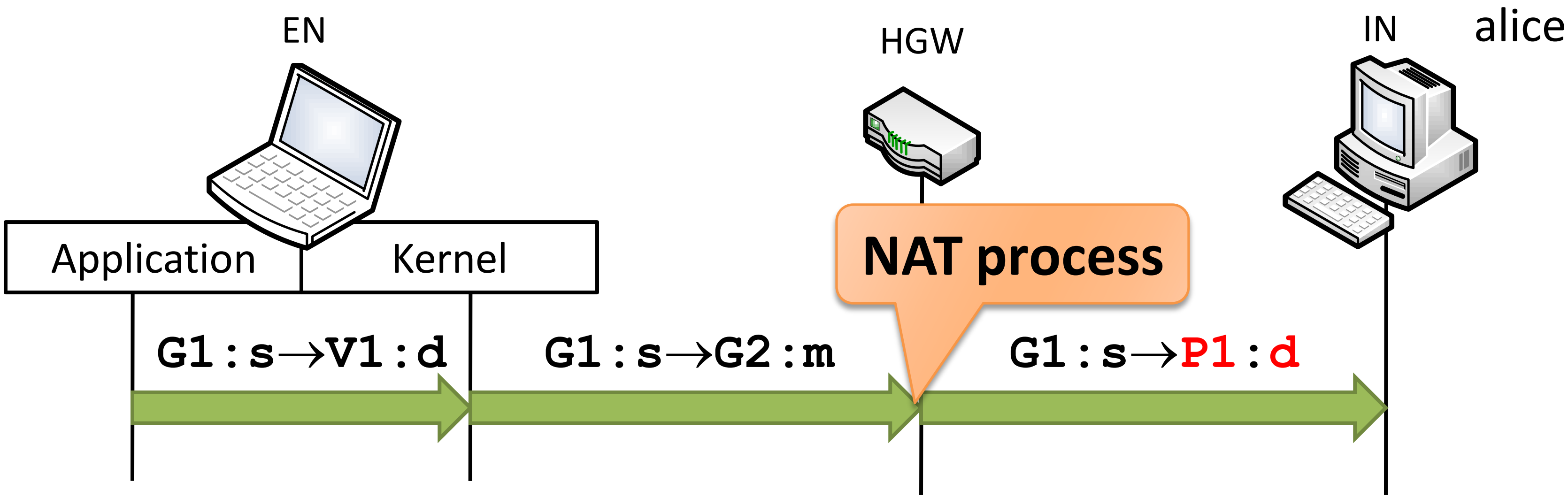


VAT table
G1:s↔{V1:d⇔G2:m}

NAT mapping
G1:s↔{G2:m⇔P1:d}

Ph.3) End-to-End Communication Based on Virtual Address Translation

- HGW handles the received packet by the normal NAT process according to the NAT mapping
  - Destination: G2:m ⇒ P1:d

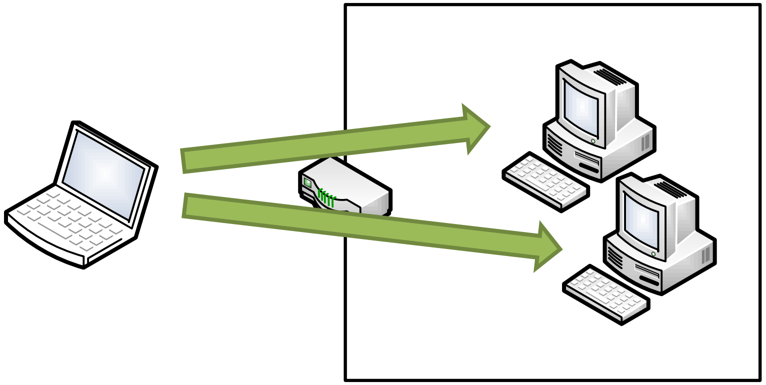


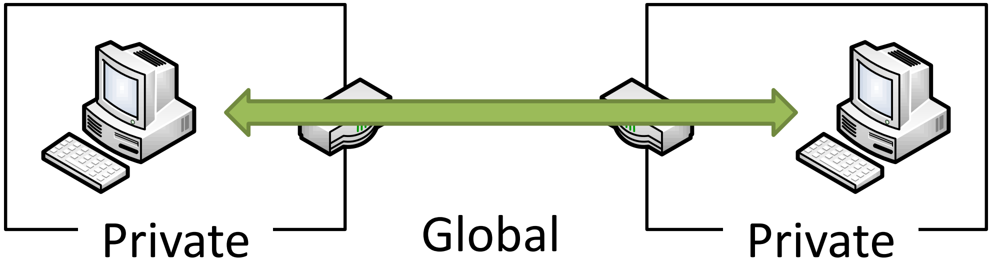
VAT table
G1:s↔{V1:d⇔G2:m}

NAT mapping
G1:s↔{G2:m⇔P1:d}

Characteristics of Our Proposed Method

- Support for:
  - TCP/UDP communication
  - All types of NATs
- A special server is not needed
  - No delay as seen in TURN
  - No single point of failure
- Flexible communication
  - Simultaneous communication with multiple INs
  - Private-to-Private communication





Conclusions

- **External Dynamic Mapping Method**
  - NAT-f is implemented in the kernel
    ➔ It is possible to use various applications
  - NAT mappings can be created from the outside of a home gateway
    ➔ INs do not need any special functions

We can communicate with any devices even if the corresponding nodes are behind the home gateway with the proposed method

Appendixes

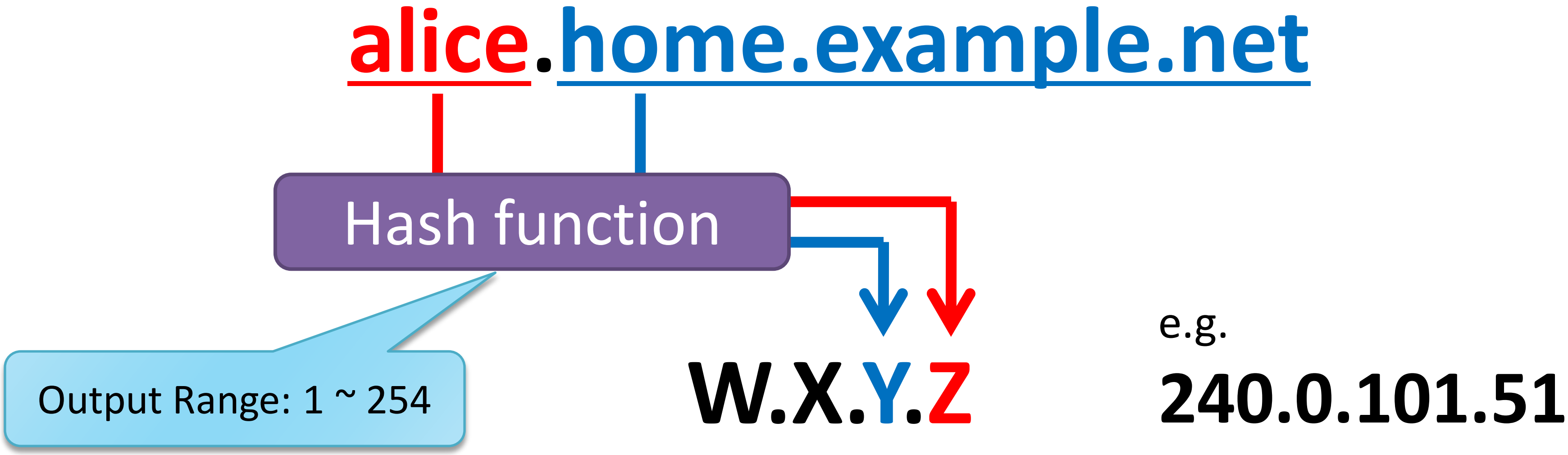
Types of Traversal Technologies

- **NAT behavior-based type:** STUN, TURN, ICE
  - Normal NAT without any modification
  - Applications on ENs and INs need to have NAT traversal functions
- **NAT control-based type:** UPnP
  - NAT mapping is created by adding functions to HGW and INs
- **NAT-less type:** 4+4, IPNL
  - All devices have to execute original process
    - e.g., routing process
  - HGW does not perform any NAT functions

Comparison with Existing Technologies

	STUN	TURN	UPnP	4+4	NAT-f
Modification to HGW	Not required		Application	Kernel	
Installation of functions on EN	Application			Kernel	
Installation of functions on IN	Application			Kernel	Not required
Special servers	Required			Not required	
Support for TCP communication	✕	○	○	○	○
Support for Symmetric NAT	✕	○	✕		○
Optimal communication route	○	✕	○	○	○

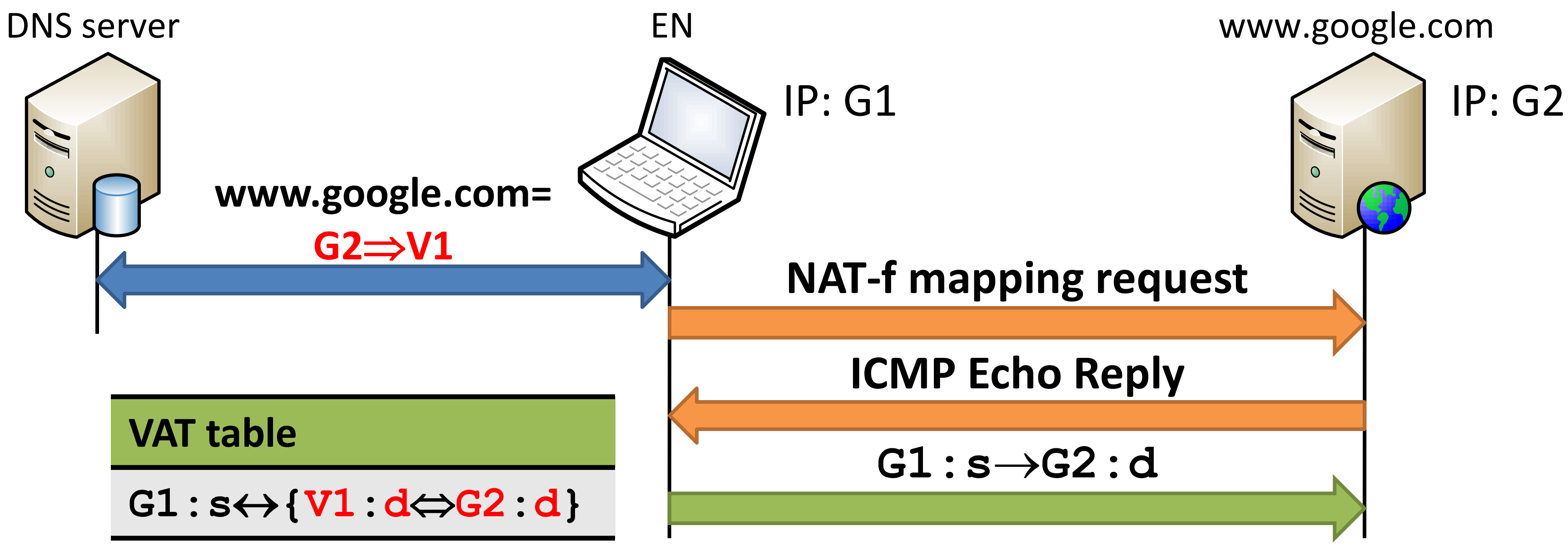
How to Define a Virtual IP Address



- **W**: Class E (user-settable value)
  - NAT-f module identifies whether it is a virtual IP address or not
- **X**: “0” (default)
  - When the hash value collies (despite of different names), change the value to prevent duplication

Which DNS Reply Is Rewritten?

- **Idea 1: All DNS reply packets**
  - If a target node does not support NAT-f, the node replies a normal ICMP Echo Reply
    - NAT-f mapping request/reply is based on ICMP Echo
  - → EN creates the VAT entry to return the virtual IP address to the original IP address of the target

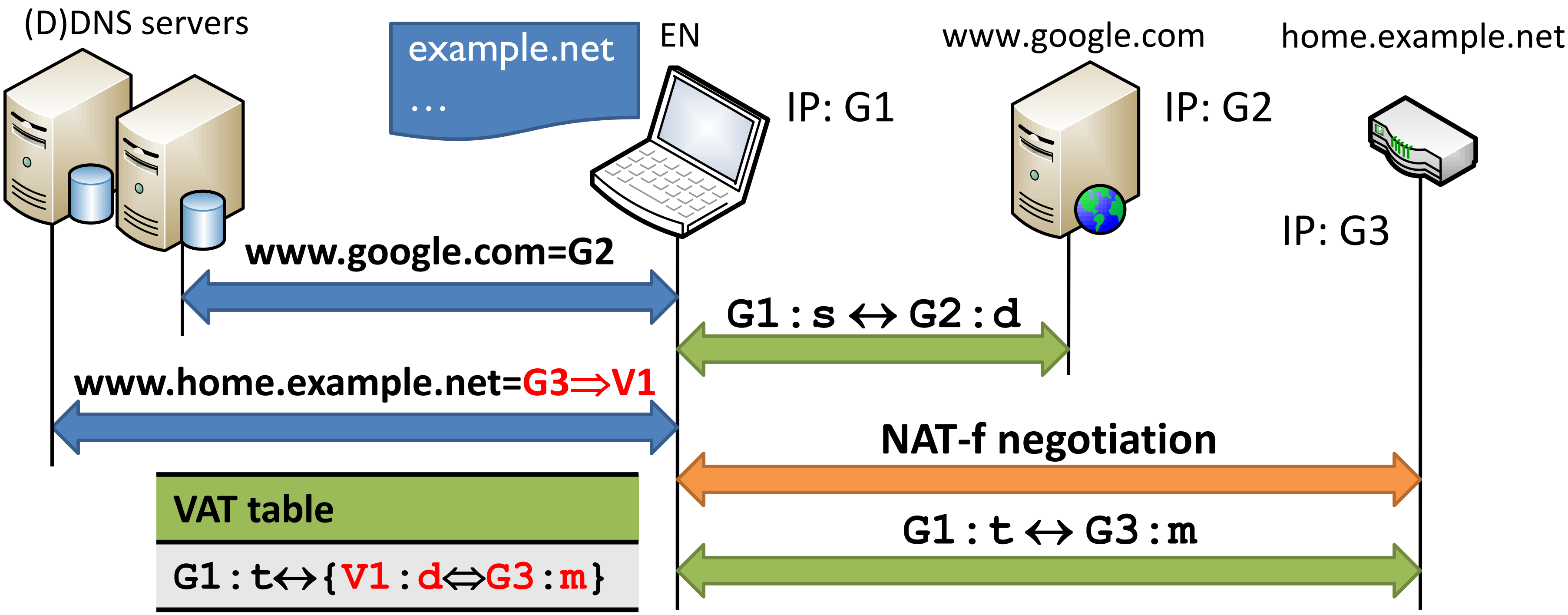


Which DNS Reply Is Rewritten?

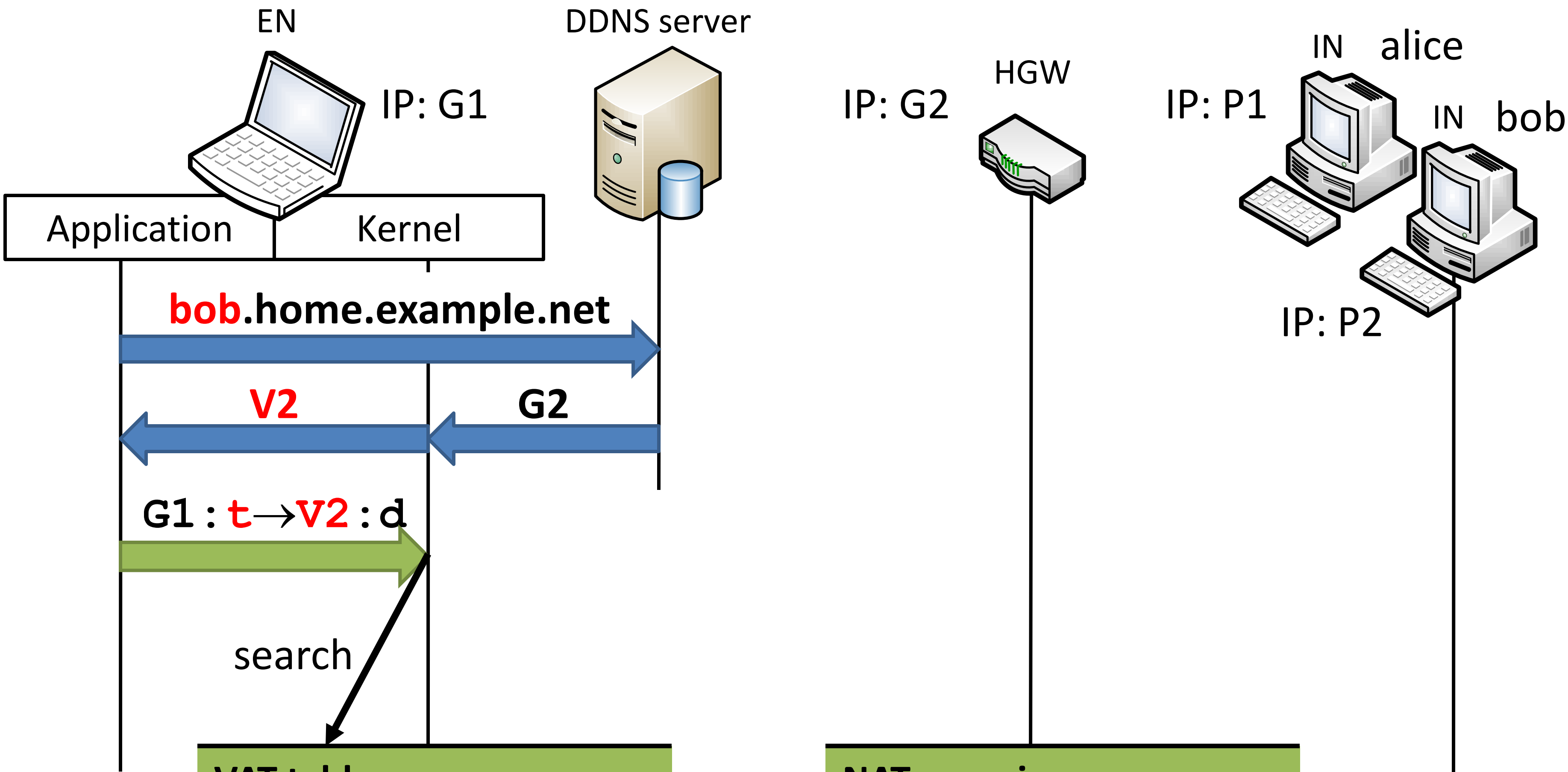
Watanabe Lab.
Department of Information Engineering, Meijo University
Information Networks Laboratory

- **Idea 2: A specific domain name**
  - EN has a domain name list of NAT-f service providers

  ➔EN rewrites the DNS reply packets registered in the list



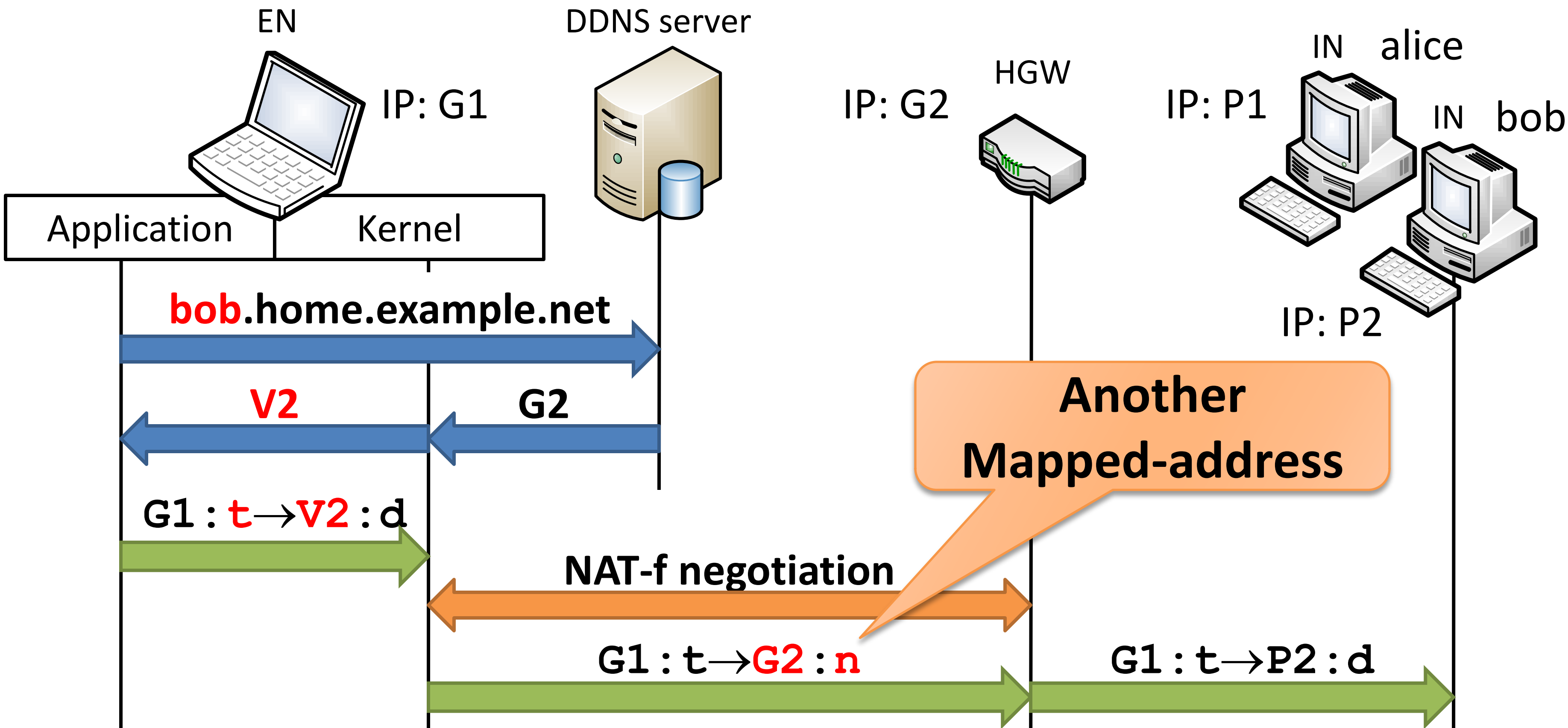
Simultaneous Communication with Multiple INs



VAT table
G1:s↔{V1:d⇔G2:m}

NAT mapping
G1:s↔{G2:m⇔P1:d}

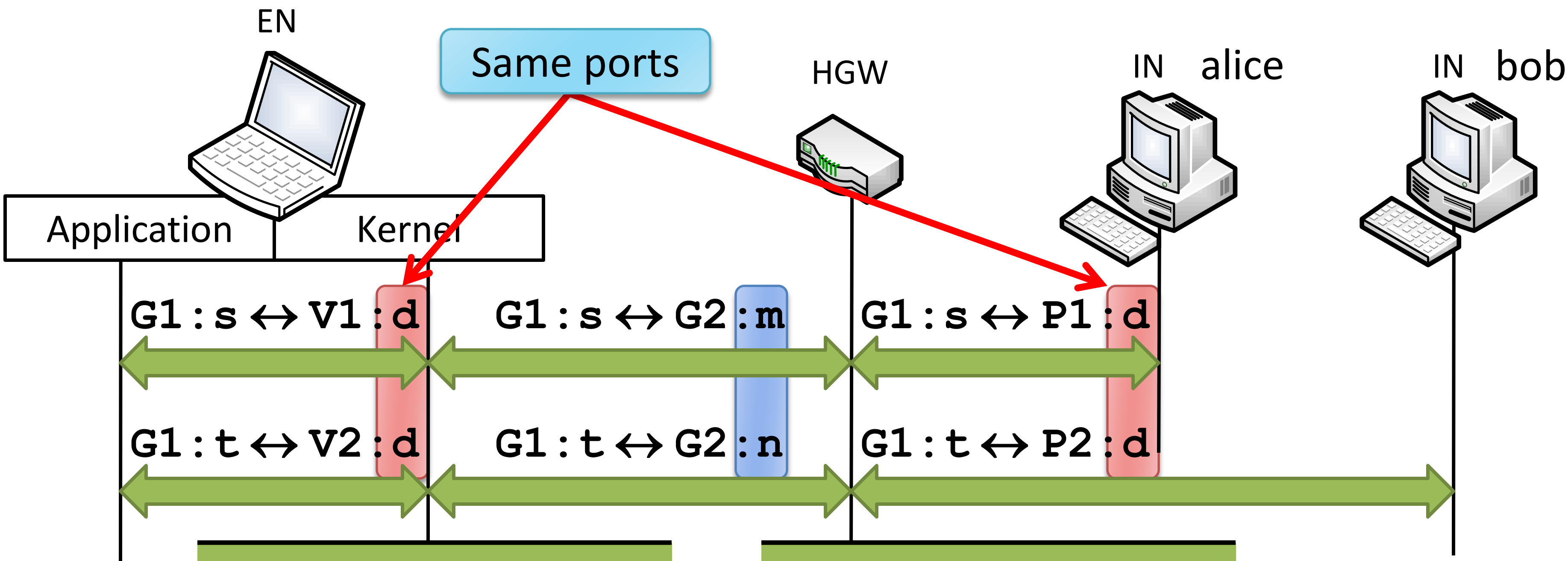
Simultaneous Communication with Multiple INs



VAT table
G1 : s↔{V1 : d⇔G2 : m}
G1 : t↔{V2 : d⇔G2 : n}

NAT mapping
G1 : s↔{G2 : m⇔P1 : d}
G1 : t↔{G2 : n⇔P2 : d}

Simultaneous Communication with Multiple INs

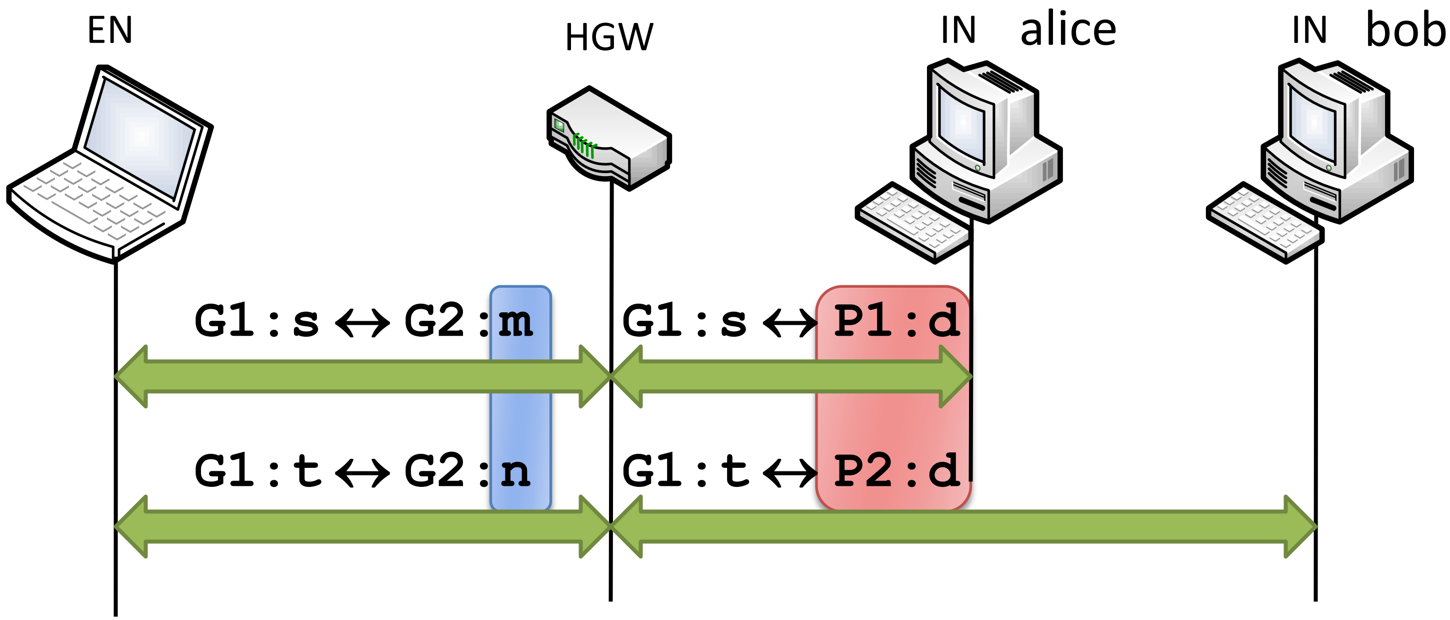


VAT table
G1:s↔{V1:d⇔G2:m}
G1:t↔{V2:d⇔G2:n}

NAT mapping
G1:s↔{G2:m⇔P1:d}
G1:t↔{G2:n⇔P2:d}

EN can communicate with multiple INs by the same destination port numbers

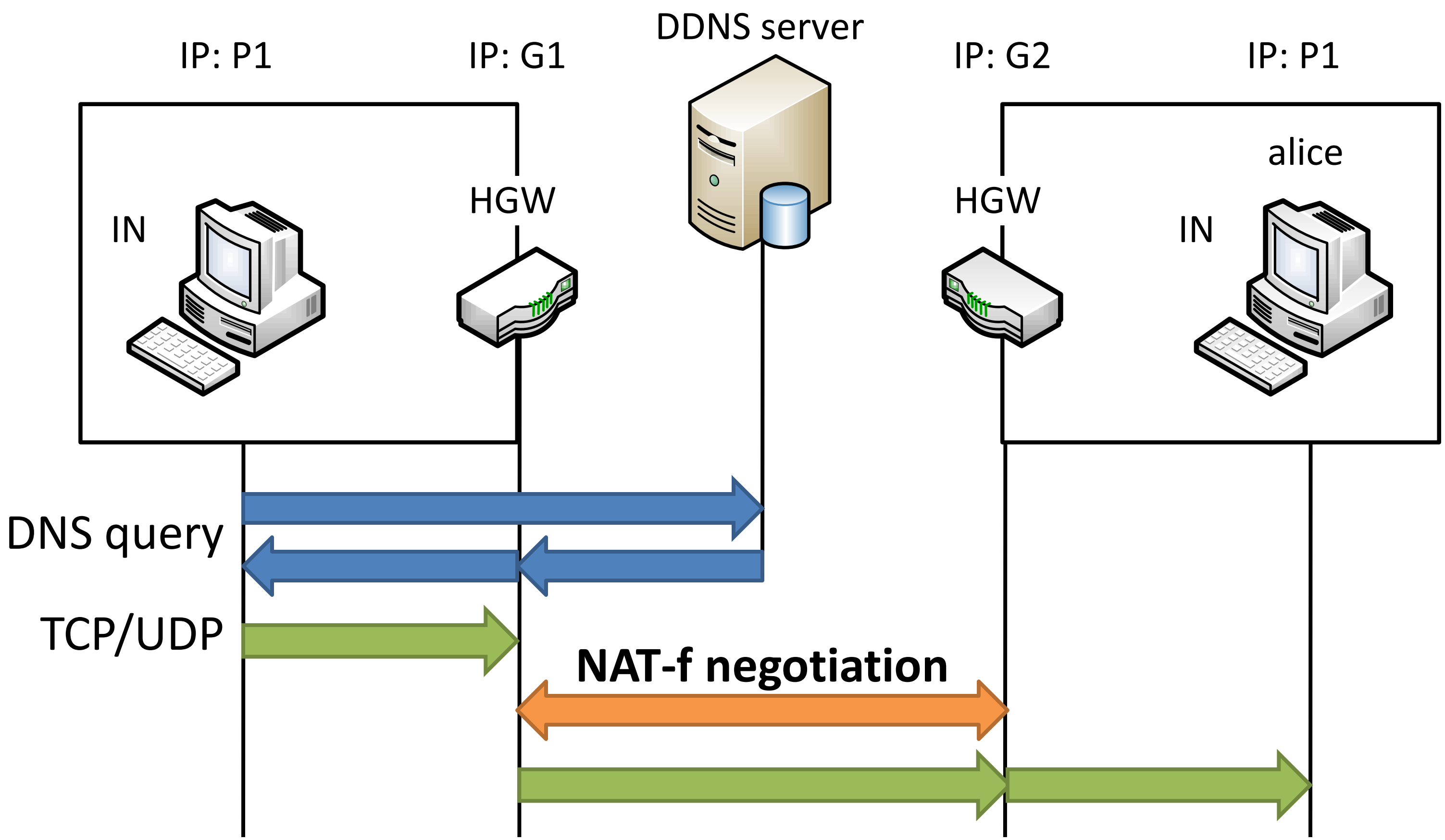
Demerits of Port-Forwarding



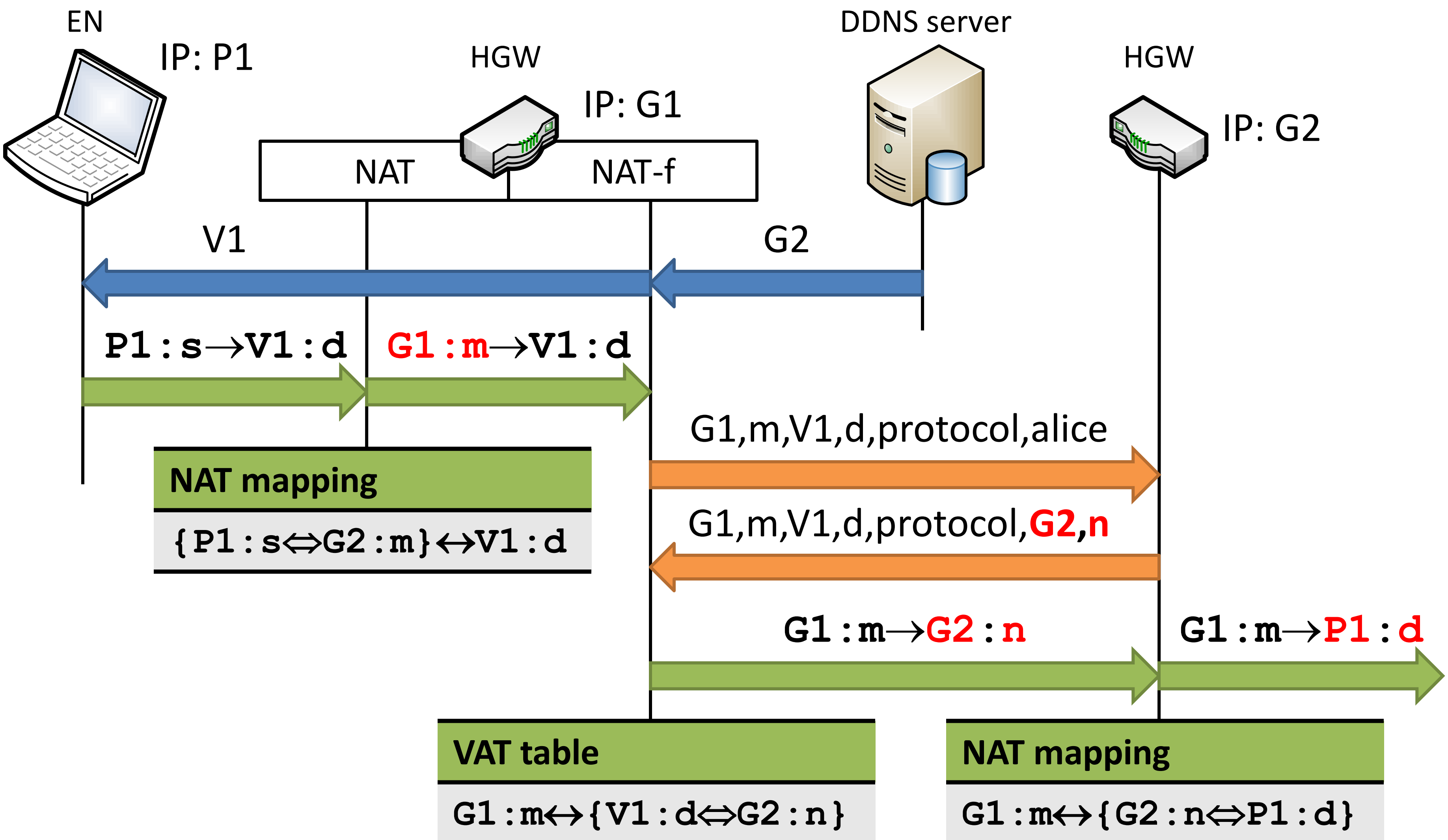
- Only a port number in IN can be related for each port number in HGW
  - A user has to configure the settings for the relations in advance
  - EN has to know the related port number in HGW

Private-to-Private Communication

- HGW implements the NAT-f module same with EN
  - NAT-f negotiation is executed between HGWs



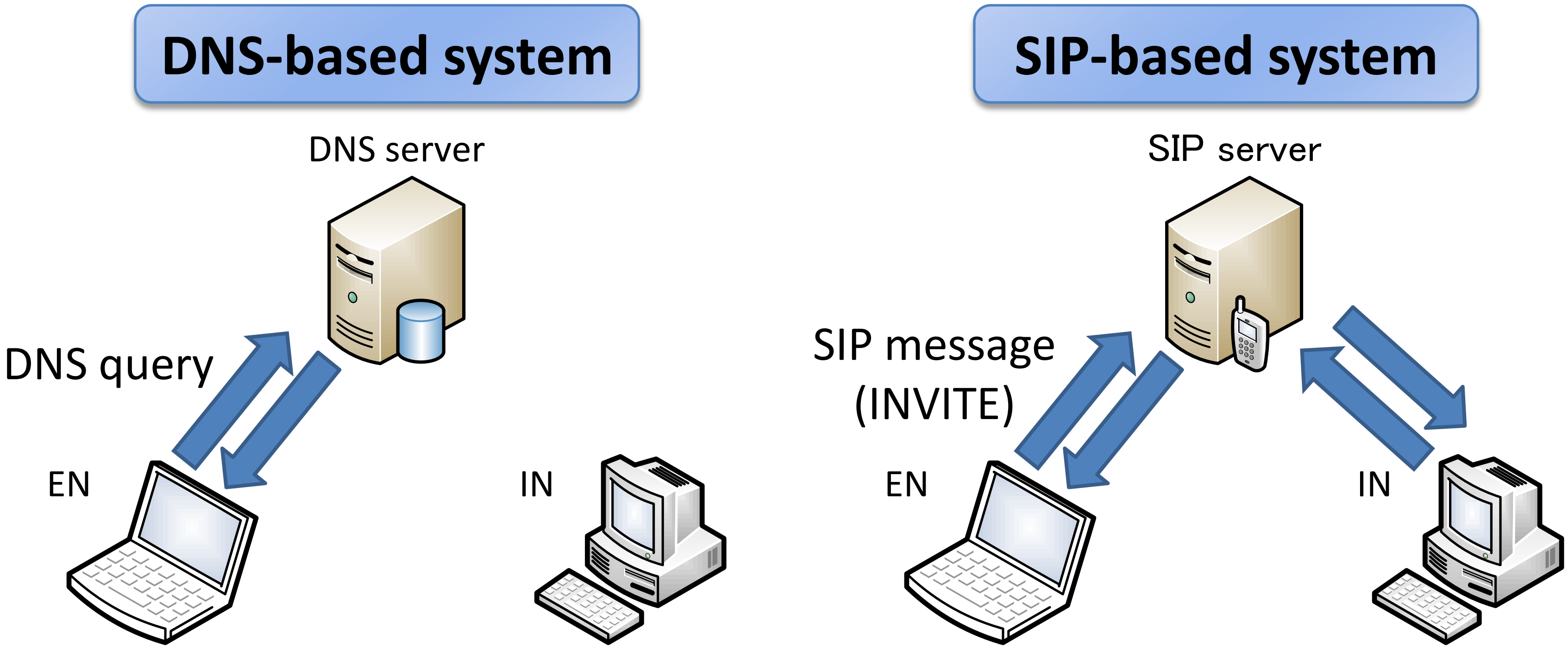
Private-to-Private Communication



Can You Use SIP Applications?

- Proposed method does not support SIP
  - Difference of the name resolution processes
  - ➔We have to extend our method or implement additional technologies for supporting SIP applications

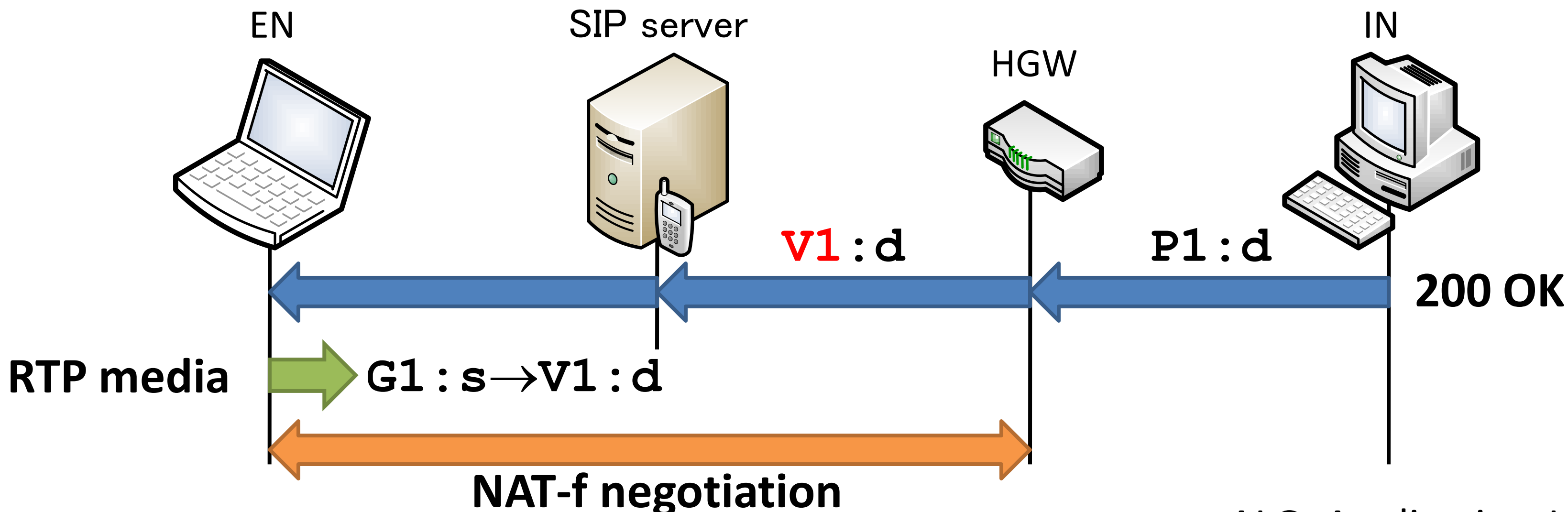
SIP: Session Initiation Protocol



Can You Use SIP Applications?

RTP: Real-time Transport Protocol

- Extending our method:
  - HGW rewrites the contents of 200 OK message
    - Private IP address of IN → Virtual IP address

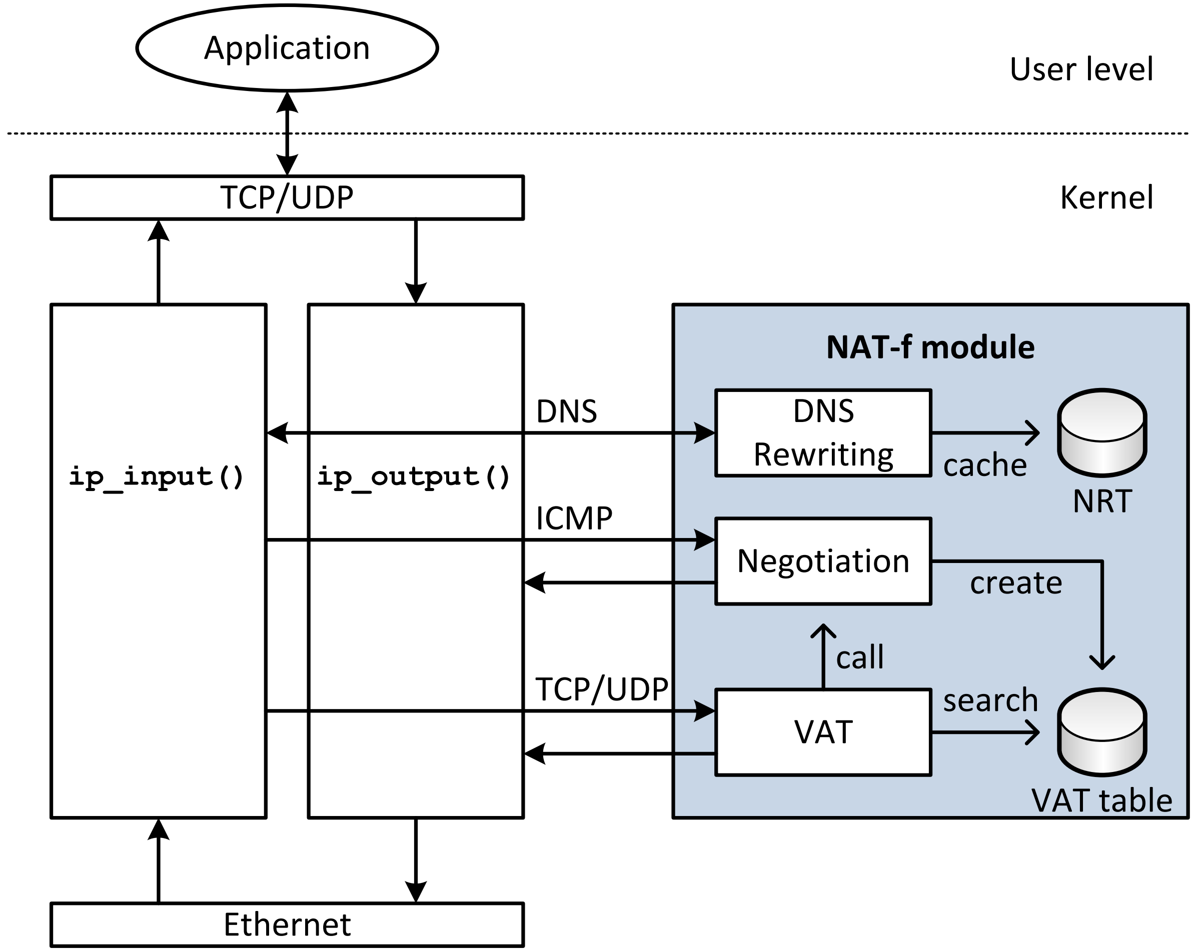


ALG: Application-Level Gateway

- Implementing additional technologies:
  - It is just needed to plug a SIP ALG into NAT/Firewall

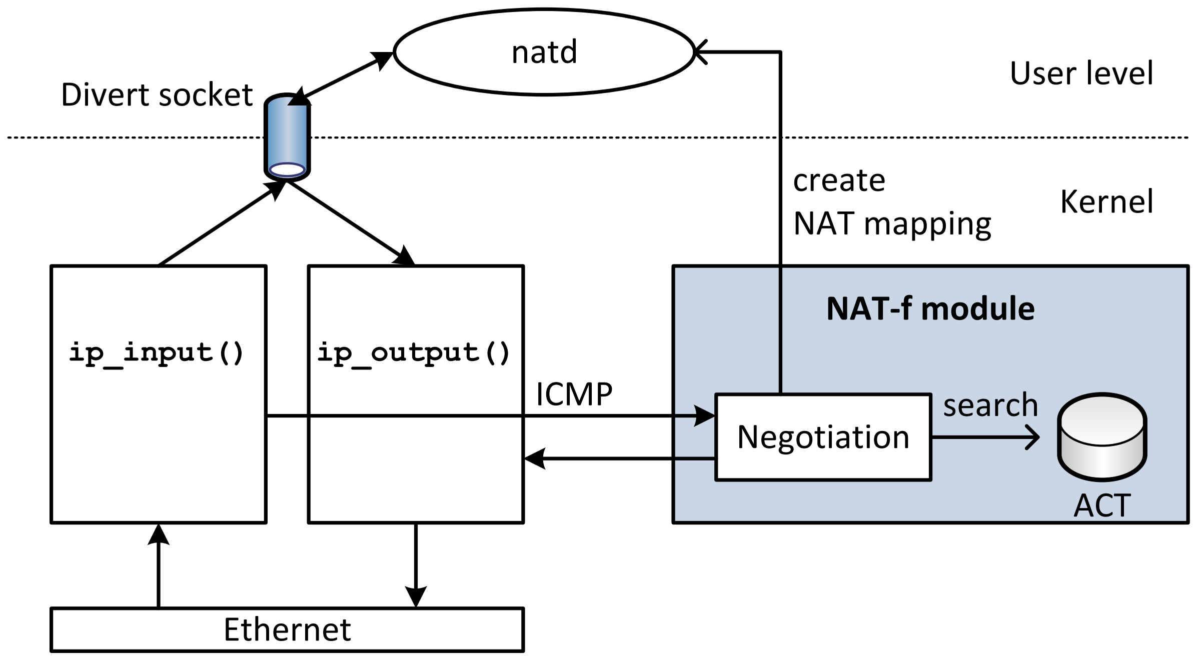
Implementation ~EN~

- NAT-f module is implemented in the IP layer on FreeBSD



Implementation ~HGW~

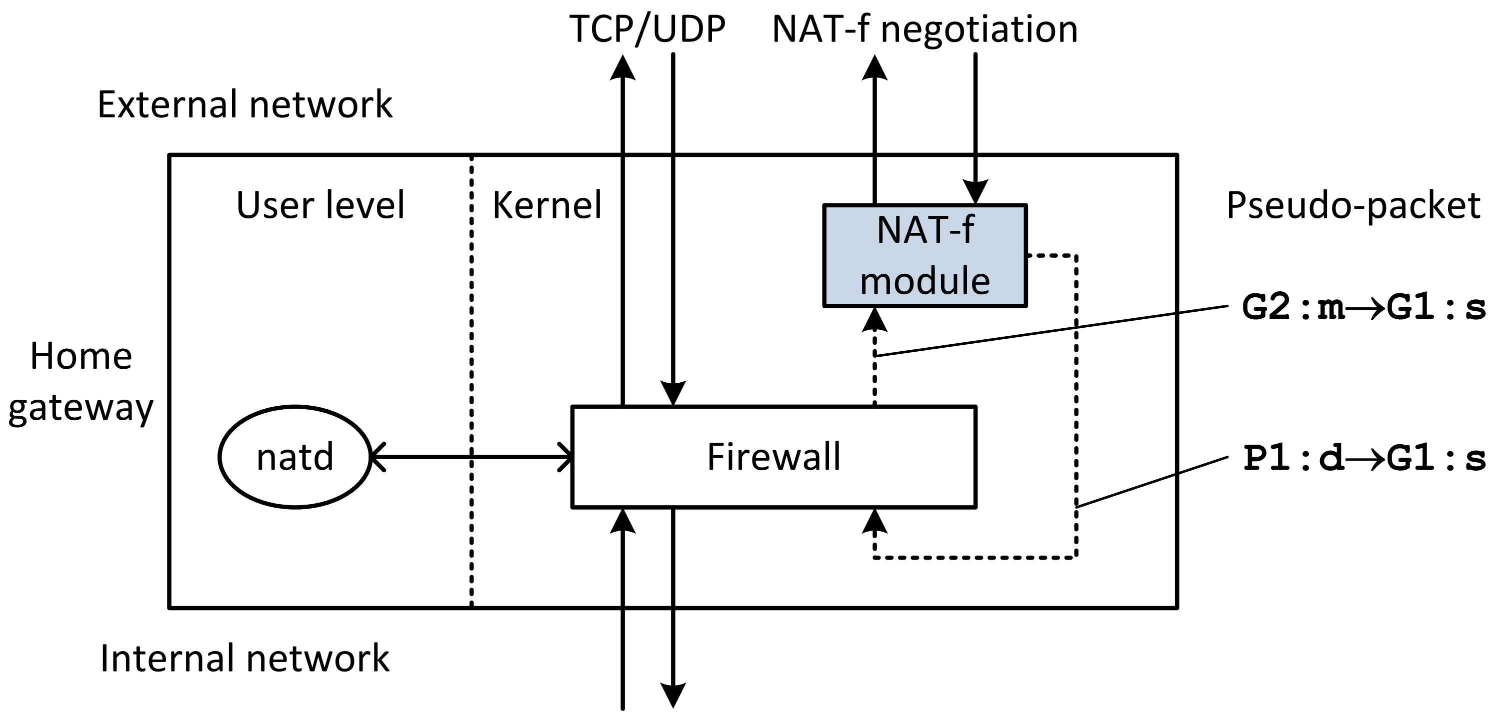
- natd: Standard NAT daemon in FreeBSD



- It is not needed to modify the kernel of HGW if the NAT-f module is implemented in “natd”

How to Create a NAT Mapping

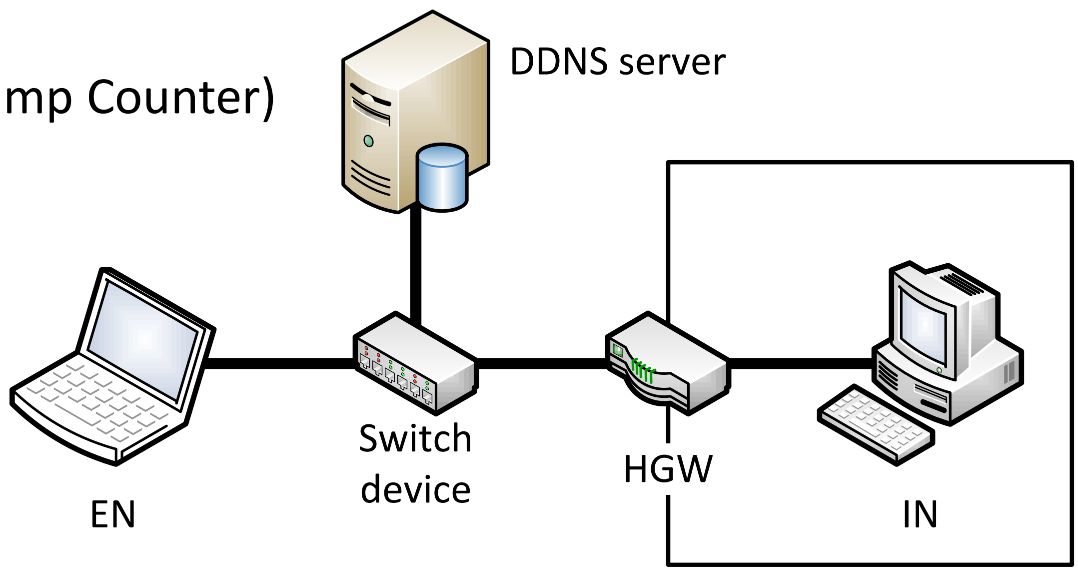
- NAT-f module makes a pseudo-packet and passes it to **`ip_input()`**
  - Source: alice (P1:d), Destination: EN (G1:s), Protocol: TCP or UDP
- natd handles it in the same process as usual



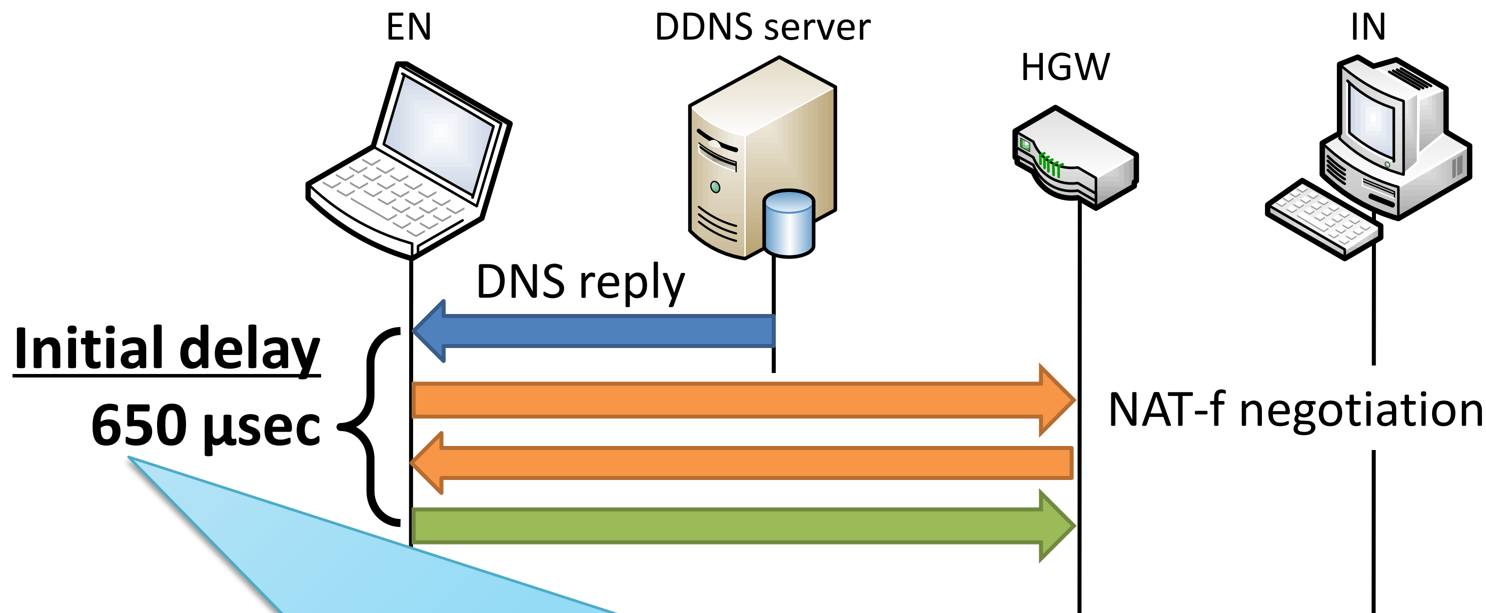
Evaluation System

- NAT-f module is implemented in the IP layer on FreeBSD 6.1-Release
- Evaluation items and tools
  - Initial delay caused by NAT-f
    - Ethereal
    - RDTSC (Read Time Stamp Counter)
  - TCP/UDP throughput
    - Netperf

Specification	
CPU	Pentium4 3.0 GHz
Memory	512 MB
NIC	100BASE-TX



Performance ~Initial Delay~



	DNS rewriting	7.1 µsec
+	NAT-f negotiation	EN: 5.1 µsec, HGW: 114.2 µsec
+	VAT process	1.1 µsec

- Our proposed method scarcely affects the TCP/UDP communication

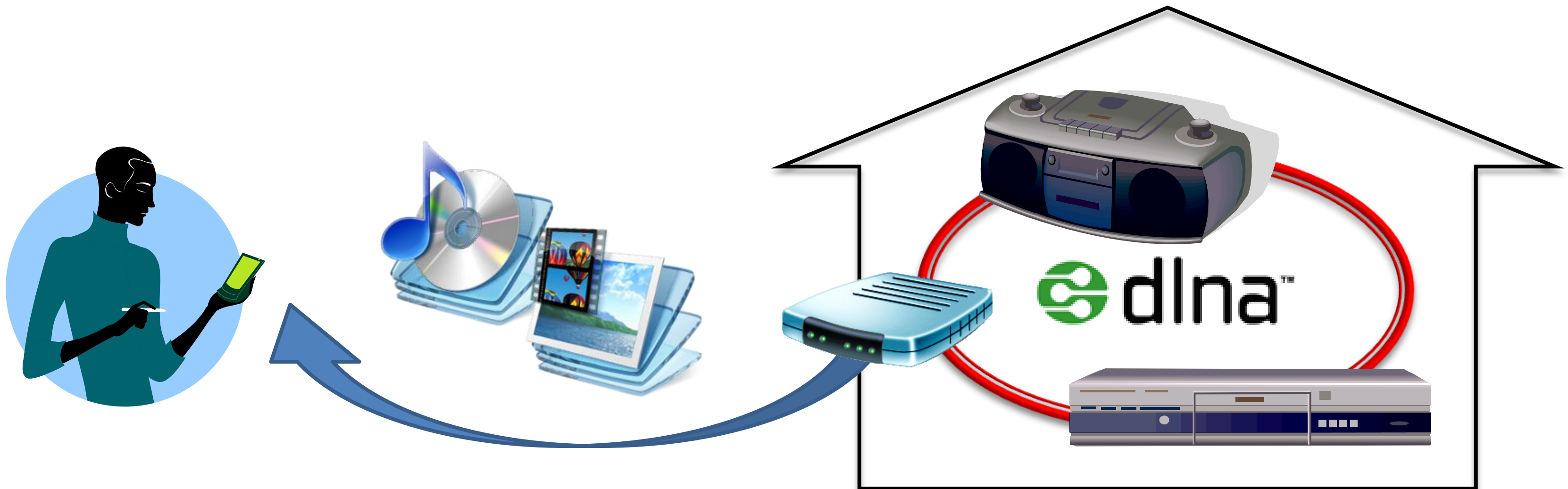
Performance ~Throughput~

Message Size (Bytes)	EN → IN TCP (Mbps)	EN ← IN TCP (Mbps)	EN → IN UDP (Mbps)	EN ← IN UDP (Mbps)
64	93.2	93.1	49.3	49.3
128	93.2	93.2	66.0	66.0
256	93.2	93.2	79.6	79.6
512	93.2	93.2	88.8	88.8
1024	93.2	93.2	94.4	96.4
1472	93.2	93.2	96.4	96.4

- Almost no difference between:
  - EN → IN : NAT-f is implemented
  - EN ← IN : NAT-f is not implemented
- VAT process executed in the EN scarcely affects the TCP/UDP throughput

Future Work

- Collaboration with DLNA (Digital Living Network Alliance)
  - A user will be able to discover and download the contents in home devices from the Internet or other home networks



- Security Considerations
  - Advanced authentication
  - Distributed Denial-of-Service attack