國立交通大學

網路工程研究所

碩 士 論 文

應用於以SIP爲基礎的VoIP環境之IP位址轉換機制

IP Address Translation Mechanisms for SIP-based VoIP

研 究 生：黃雅琳

指導教授：林一平　教授

中 華 民 國　九 十 六　年　七　月

應用於以SIP爲基礎的VoIP環境之IP位址轉換機制
IP Address Translation Mechanisms for SIP-based VoIP

研 究 生：黃雅琳　　Student：Ya-Lin Huang

指導教授：林一平　　Advisor：Yi-Bing Lin

國 立 交 通 大 學

網 路 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Network Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

July 2007

Hsinchu, Taiwan, Republic of China

中華民國 九十六 年 七 月

應用於以 SIP 爲基礎的 VoIP 環境之 IP 位址轉換機制

學生：黃雅琳　　指導教授：林一平　教授

國立交通大學網路工程研究所碩士班

摘　　要

在以 Session Initiation Protocol (SIP) 爲基礎之 Voice over IP (VoIP) 服務中，網路電話使用 IP 位址表示其所在位置，因此 IPv4 位址不足的問題會影響 SIP 多媒體服務的佈建。目前常見的解決方法是，在 IPv4 網路中佈建網路位址轉換器 (Network Address Translator; NAT)，使得多支網路電話共用一個 IPv4 位址，但如此卻導致 SIP/Real-time Transport Protocol (RTP) 穿越 NAT 的問題發生。爲此我們研發出支援現有解決穿越 NAT 問題方法的網路電話軟體，並討論各種解決方法的優缺點。而另一解決 IPv4 位址不足問題的方法是採用 IPv6 來提供足夠的位址，使得每支網路電話均有其獨一無二的 IPv6 位址。只是在佈建 IPv6 初期，必須面臨 IPv4 與 IPv6 互通而產生的問題。對於此問題，我們提出一有效率的解決方法，其在通話建立以及語音傳送的效能表現皆優於現有方法。根據我們的分析，穿越 NAT 的問題尚未有十全十美的解決方法，而使用 IPv6 搭配我們所提出解決 IPv4 與 IPv6 互通問題的方法，對於佈建以 SIP 爲基礎之 VoIP 服務是最有效率的方案。

關鍵字： ICE, IPv4/IPv6 互通, NAT, 重新導向, SBC, SIP, SIP-ALG, STUN, IP 轉換, UPnP, VoIP, VPN

IP Address Translation Mechanisms for SIP-based VoIP

Student: Ya-Lin Huang Advisor: Dr. Yi-Bing Lin

Institute of Network Engineering
National Chiao Tung University

ABSTRACT

In Session Initiation Protocol (SIP)-based Voice over IP (VoIP) services, IP addresses are used as location information of User Agents (UAs). The insufficiency of IPv4 addresses becomes a problem in SIP-based VoIP deployment. Deploying Network Address Translators (NATs) into IPv4 networks is one approach so that a public IPv4 address is shared by multiple UAs. However, this approach leads to SIP/Real-time Transport Protocol (RTP) NAT traversing problem. We design and develop a UA that supports the existing SIP/RTP NAT traversing solutions. This UA is used for further analyses and experiments. Another approach is to adopt IPv6 so that every UA owns a unique IPv6 address. Nevertheless, IPv4/IPv6 interworking problem for SIP-based VoIP may occur in the early stage of IPv6 deployment. To solve this problem, we propose an effective solution where the least call setup overhead is introduced and no impairment is incurred to the RTP transmission performance. Our study indicates that none of the existing SIP/RTP NAT traversing solutions is perfect in all aspects. Instead, the usage of IPv6 cooperating with our proposed solution is the best choice with the most efficient performance for deploying SIP-based VoIP services.

Keywords: ICE, IPv4/IPv6 interworking, NAT, redirect, SBC, SIP, SIP-ALG, STUN, translation, UPnP, VoIP, VPN

Acknowledgements

I would like to express my sincere thanks to my advisor, Prof. Yi-Bing Lin. Not only perspicacious advice and strict training in research but also valuable suggestion and timely encouragement regarding future career and philosophy of life, he provides. Also, he is always patient of my slow improvement and never gives me up. Without his supervision, I will not have a chance to complete this thesis and to be who I am.

Next, I feel grateful for endless help from seniors: thanks to my master committee members, Dr. Jeu-Yih Jeng, Dr. Yuan-Kai Chen, Prof. Wei-Ru Lai, Prof. Wei-Zu Yang, Prof. Phone Lin, Prof. Ai-Chun Pang, Prof. Shun-Ren Yang, Dr. Hsien-Ming Tsai, Prof. Quincy Wu, and Dr. Whai-En Chen, for their practical comments; thanks to Prof. Rong-Jaye Chen and Prof. Ying-Dar Lin for their recommendation towards higher education and scholarships; thanks to Prof. Ying-ping Chen for his recommendation as well as a great deal of living philosophy and experiences; thanks to senior members in Lab 117 for their kind sharing and beneficial guidance, especially to Prof. Pei-Chun Lee, Dr. Lin-Yi Wu, Ya-Chin Sung, and Meng-Hsun Tsai.

Also, I appreciate inexhaustible support from my lovely friends, including my high schoolmates, university colleagues, and all the others known through common interests, friends, and the Internet. Thank them for priceless suggestion and heartily honesty. Particularly, I would like to mention Bing-Jhen Wu, Hui-Chen Lin, and Ya-Ching Chiu for their thoughtfulness all the time, especially when I was in a terrible status.

Last but not least, I owe a great deal to my parents and my elder brother. Thank them for providing me with a perfect environment to grow up physically and mentally; thank them for instructing me in correct values; thank them for giving me freedom to develop my personality. Most important of all, thank them for always being supportive even though they did not really quite understand what I was working for. I feel truly lucky and satisfied to be their family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Session Initiation Protocol (SIP) [23] is a Voice over IP (VoIP) signaling protocol for establishing calls, where the voice and the multimedia data are typically transmitted using Real-time Transport Protocol (RTP) [24]. The information of the voice and the multimedia data (e.g., the data type, the media codec, and the IP address/port number that the data should be sent to) is carried in the SIP message body, and mostly it is encoded in the format of Session Description Protocol (SDP) [16] [18]. In SIP, User Agents (UAs) are the IP network endpoints just like telephones in the telephone networks. Since every UA requires a unique IP address to serve as location information, the shortage of IPv4 addresses becomes a problem in SIP deployment. This problem can be solved by either introducing Network Address Translators (NATs) [27] into IPv4 networks or replacing IPv4 with IPv6 [13].

With an NAT connecting a private network to a public network (e.g., the Internet), a public IPv4 address is shared by multiple UAs in the private network. However, a UA cannot establish a call correctly when it resides in the private network. This issue is referred to as *SIP/RTP NAT traversing problem*. Several solutions have been proposed in the literature. According to our analyses, none of them is perfect in all aspects.

On the other hand, IPv6 provides large address space so that every UA owns a unique IPv6 address. In other words, NAT no longer exists, and neither does SIP/RTP NAT traversing problem. Nevertheless, in the early stage of IPv6 deployment, IPv6 UAs and IPv4 UAs coexist in the same network. Without proper mechanisms, IPv6 and IPv4 UAs cannot interact with each other correctly. This issue is referred to as *IPv4/IPv6 inter-*

working problem for SIP-based VoIP. Three solutions have been proposed in the literature, but each of them has disadvantages. Therefore, we propose an effective solution where the least extra call setup overhead is introduced and no impairment is incurred to the RTP transmission performance. The rest of this chapter introduces the background information of the IP-related SIP/SDP fields and the network address translation mechanism.

1.1 IP-related SIP/SDP Fields

Several SIP header fields in a SIP message contain the IP address and the port number (*transport address*) for SIP message delivery. For example,

- The *Via* header fields indicate the SIP nodes visited by a SIP request so far. The reverse direction of the path should be followed to route the SIP responses for this request. When a SIP request is received, the IP address in the topmost *Via* header field is compared to the IP address in the *Source Address* header of the IP packet that carries the SIP request (*source IP address*). If they are different, a *received* parameter containing the source IP address is added to the topmost *Via* header field. In this case, a SIP response must be sent to the IP address specified in the *received* parameter of the topmost *Via* header field, instead of the IP address in the topmost *Via* header field.

- The *Contact* header field indicates the transport address where the other party can send subsequent SIP requests.

Two SDP fields in the SIP message body provide the transport address for the media session.

- The IP address is provided in the *c* field.

- The port number is provided in the *m* field.

1.2 Network Address Translation Mechanism

Figure 1.1 shows the NAT architecture. In this figure, a private network (Figure 1.1 (1)) connects to the Internet (Figure 1.1 (2)) through an NAT. The private IP addresses

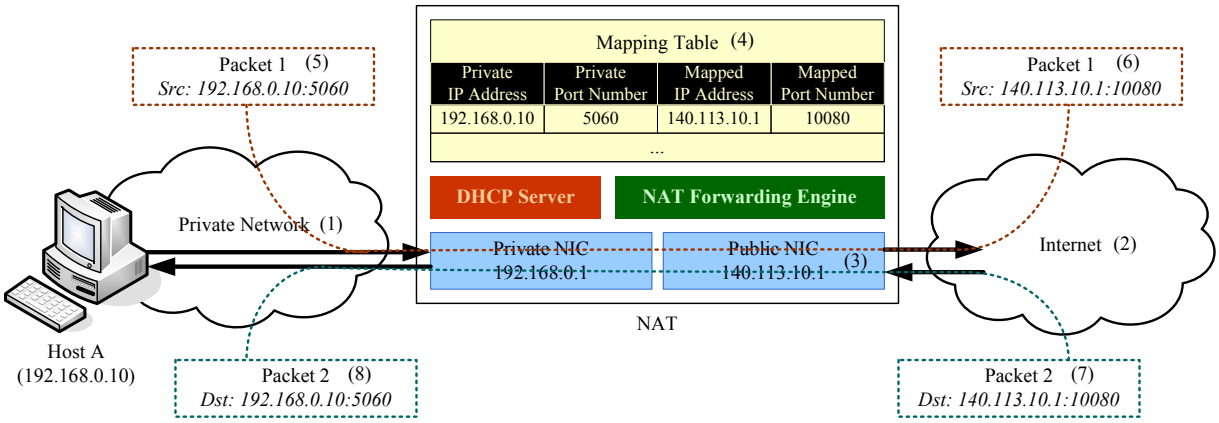


Figure 1.1: NAT Architecture

192.168.0.0/24 are assigned to the hosts in the private network. The IP address of the public Network Interface Card (NIC, Figure 1.1 (3)) in the NAT is 140.113.10.1. Suppose that Host A sends a packet (Packet 1 with the source transport address 192.168.0.10:5060; see Figure 1.1 (5)) to the destination in the Internet. In most cases, the source transport address of the packet is treated as the destination transport address of another packet that is sent back to Host A. The packet with the private destination IP address 192.168.0.10 cannot be routed in the Internet. Therefore, the NAT performs the transport address translation at the network and the transport layers before sending Packet 1 to the Internet. Specifically, the NAT replaces the source IP address by the public IP address of the NAT (i.e., 140.113.10.1; see Figure 1.1 (3)) and changes the source port number to an unused port number 10080 in the NAT (see Figure 1.1 (6)). The mapping between the private transport address and the public transport address (*private-to-public transport address mapping*) is stored in the NAT's mapping table (Figure 1.1 (4)). Packet 1 is then routed to the destination in the Internet. When the NAT receives Packet 2 (a packet with the destination transport address 140.113.10.1:10080 from the Internet; see Figure 1.1 (7)), it retrieves the mapping from the mapping table to translate the destination transport address into 192.168.0.10:5060 and then sends Packet 2 to Host A (Figure 1.1 (8)). Note that the NAT only translates the transport address at the network and the transport layers. The application-layer content is left unchanged.

The rest of this thesis is organized as follows. Chapter 2 describes SIP/RTP NAT traversing problem in detail. Also, the existing solutions of the problem are discussed and

analyzed. Chapter 3 elaborates IPv4/IPv6 interworking problem for SIP-based VoIP as well as the existing solutions. Besides, an effective solution is proposed and compared to those existing solutions. Finally, this thesis is concluded in Chapter 4.

Chapter 2

SIP/RTP NAT Traversing

An NAT only translates the transport address at the network and the transport layers. It does not translate the transport address carried in the content of a SIP message. This results in the inconsistency between the transport address at the application layer and that at the network and the transport layers when the SIP message traverses the NAT. This issue is further elaborated in the following example.

Figure 2.1 illustrates SIP message delivery between UA1 (in the private network) and UA2 (in the Internet) through a standard NAT. UA1 is assigned a private IP address 192.168.0.10, and UA2 is assigned a public IP address 140.113.10.10. As in Figure 1.1, the NAT has two IP addresses: 192.168.0.1 for the private NIC and 140.113.10.1 for the public NIC. Suppose that UA1 sends a SIP INVITE message (Figure 2.1 (1)) to UA2. In this message, both the *Via* and the *Contact* header fields contain UA1's IP address 192.168.0.10 and the port number 5060. For the RTP media session, 192.168.0.10 and 9000 are recorded in the *c* and the *m* fields, respectively. This message is carried by an IP packet with the source transport address 192.168.0.10:5060. At the NAT, the source transport address is translated to 140.113.10.1:10080 (Figure 2.1 (2)). However, the application-layer content (i.e., the SIP message) is left unchanged.

Upon receipt of the INVITE message, UA2 creates a SIP 200 OK message where the *Via* header field (i.e., 192.168.0.10:5060) is copied from the INVITE message. Then UA2 adds a *received* parameter with the value 140.113.10.1 to the *Via* header field. UA2 replies the 200 OK message using the IP address 140.113.10.1 and the port number 5060 in the *Via* header field (Figure 2.1 (3)). Since 5060 is not a correct port number in the NAT's

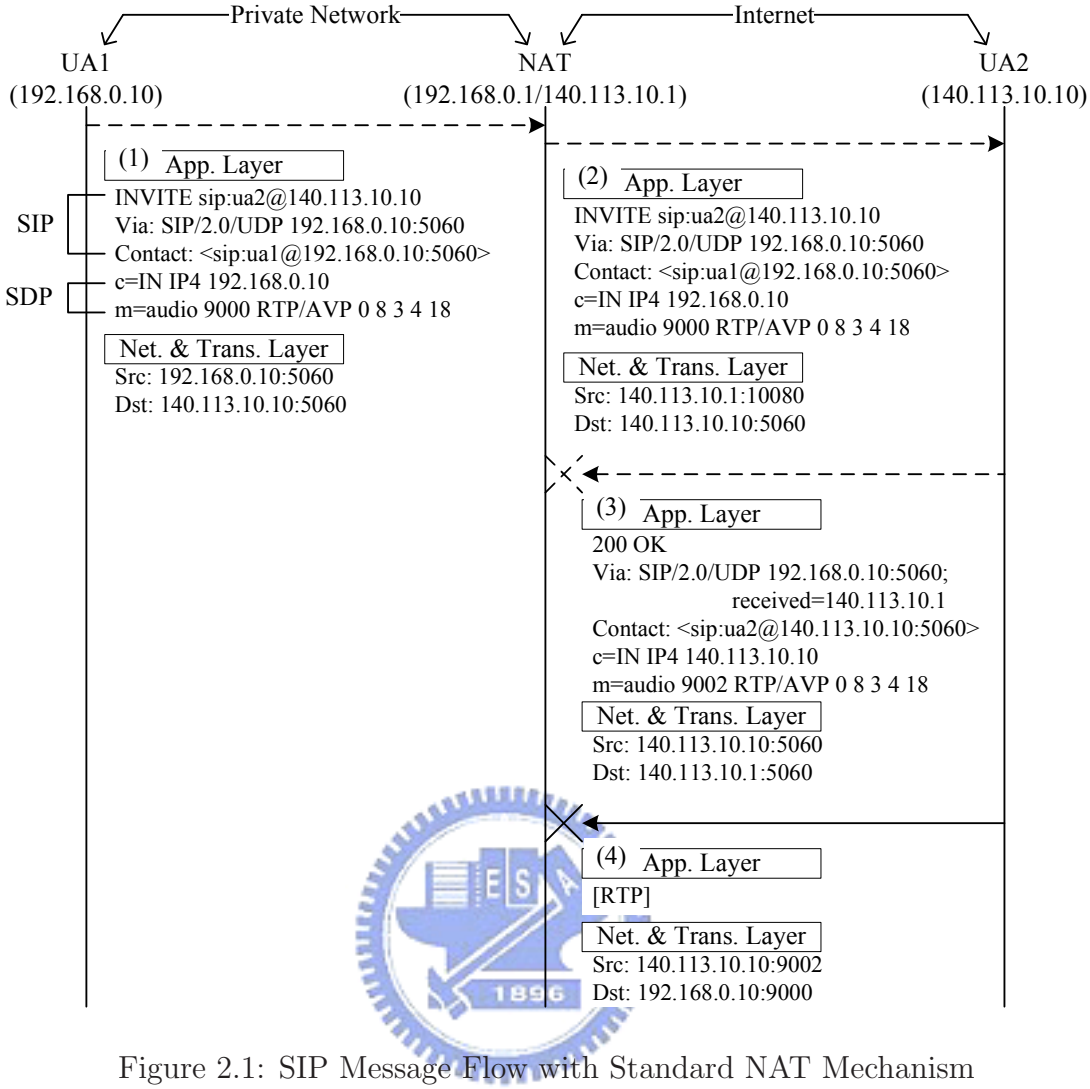


Figure 2.1: SIP Message Flow with Standard NAT Mechanism

mapping table (10080 is the correct port number), this message cannot be delivered to the destination (i.e., UA1). Also, the RTP packets are delivered to 192.168.0.10:9000 (Figure 2.1 (4)) as designated by the *c* and the *m* fields in the INVITE message. Clearly, the destination is unreachable from the Internet.

The SIP/RTP NAT traversing problem consists of three main scenarios where the UA in the private network performs registration, call origination, and call termination. The UA performs registration in order to announce its location information so that other UAs can reach it. Therefore, the private transport address in the *Via* and the *Contact* header fields becomes a problem in registration. In call origination, not only the private transport address in the *Via* and the *Contact* header fields but also that in the *c* and the *m* fields cause the SIP/RTP NAT traversing problem. In call termination, the UA

should be reached first and then establishes the call. In other words, the UA in the private network performs correct call termination if the SIP/RTP NAT traversing problems in registration and call origination are solved. Since most SIP/RTP NAT traversing solutions for call origination can be applied to registration, this thesis focuses on the call origination scenario.

The SIP/RTP NAT traversing problem can be resolved by two types of solutions: *UA-based* and *server-based*. In the UA-based solution, the application-layer transport address translation is performed at the UA in the private network. In the server-based solution, the translation is performed at a server in the Internet. Note that in the UA-based solution, the UA may still need to interact with a server to obtain the specific information for a particular solution. Examples of UA-based solutions include Virtual Private Network (VPN) [7], Static Route [2], Universal Plug and Play (UPnP) [4], Simple Traversal of UDP through NATs (STUN) [20], STUN Relay Usage [21], Interactive Connectivity Establishment (ICE) [19], and Realm Specific IP (RSIP) [8]. Examples of server-based solutions include SIP-Application Layer Gateway (SIP-ALG) [11], Session Border Controller (SBC) [9], and midcom [28]. This thesis focuses on several widely used SIP/RTP NAT traversing solutions (i.e., VPN, Static Route, UPnP, STUN, ICE, SIP-ALG, and SBC) and shows their trade-offs.

2.1 The VPN Solution

Figure 2.2 illustrates the VPN solution [7], where UA1 (Figure 2.2 (1)) is a VPN client in the private network while both UA2 (Figure 2.2 (2)) and the VPN server (Figure 2.2 (3)) reside in the Internet. The IP address settings for UA1, UA2, and the NAT are the same as those in Figure 2.1, and the IP address of the VPN server is 140.113.20.1. The VPN server maintains an address pool with IP addresses ranging from 140.113.20.2 to 140.113.20.254. After UA1 has established a VPN tunnel (Figure 2.2 (5)) connecting to the VPN server, a virtual NIC with a public IP address 140.113.20.10 (belonging to the VPN server) is created at UA1. This virtual NIC is set as the default interface. That is, the public IP address is used for all outgoing packets from UA1. On the other hand, all

packets from the Internet with the destination IP address 140.113.20.10 are first sent to the VPN server, and the VPN server forwards these packets to UA1 through the VPN tunnel.

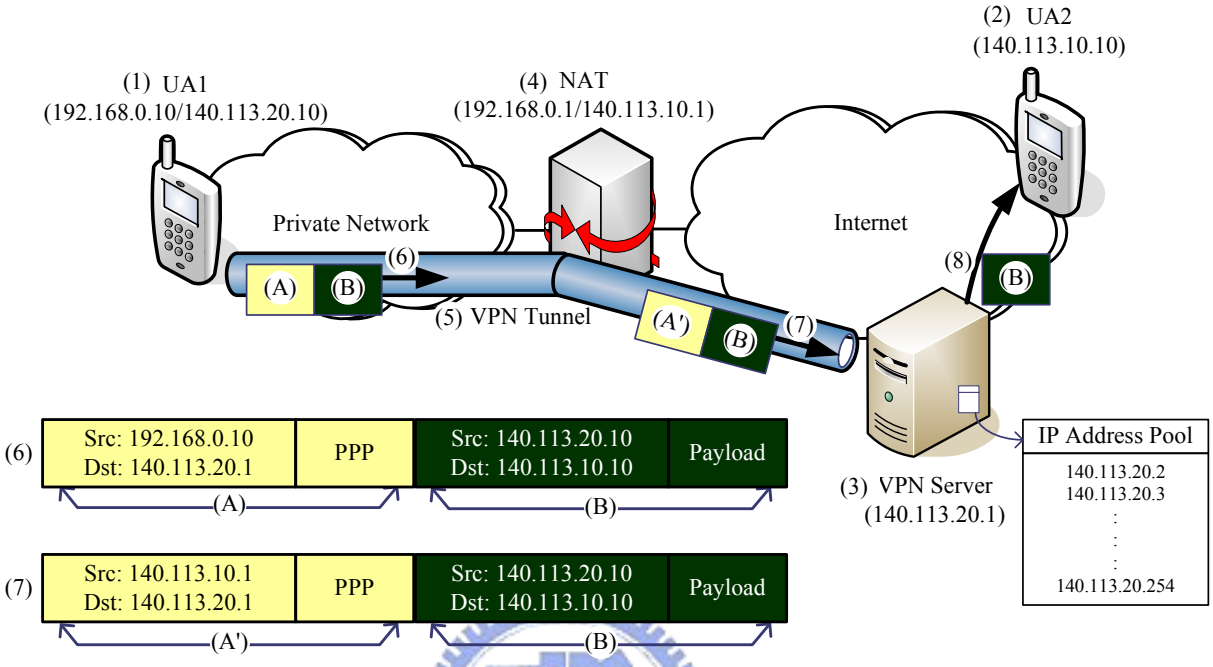


Figure 2.2: SIP/RTP NAT Traversing: the VPN Solution

When UA1 sends a SIP message to UA2, the transport address 140.113.20.10:5060 is filled in the *Via* and the *Contact* header fields. Also, 140.113.20.10 and 9000 are filled in the *c* and the *m* fields, respectively. This message is then carried by an IP packet with the source IP address 140.113.20.10 (Figure 2.2 (B)). In order to deliver the packet through the VPN tunnel, the packet is encapsulated by the IP and the Point-to-Point Protocol (PPP) [6] (Figure 2.2 (A)) to be a tunneled packet. Before the tunneled packet arrives at the NAT (Figure 2.2 (6)), the source and the destination IP addresses are 192.168.0.10 and 140.113.20.1, respectively. After the tunneled packet leaves the NAT (Figure 2.2 (7)), the source IP address is translated to 140.113.10.1 by the NAT. The tunneled packet is then sent to the VPN server. Upon receipt of the tunneled packet, the VPN server decapsulates the packet and sends it to UA2 (Figure 2.2 (8)). Since the public IP address 140.113.20.10 (of the VPN server) is contained in the SIP message received by UA2, UA1 is reachable from the Internet through the VPN tunnel, as described in the following example.

Table 2.1: Mapping Table in the NAT and the UA (A Simplified Version)

Index	Private IP Address	Private Port Number	Mapped IP Address	Mapped Port Number
1	192.168.0.10	5060	140.113.10.1	10080
2	192.168.0.10	9000	140.113.10.1	19000

When UA2 replies a SIP message to UA1, the message is carried by an IP packet with the destination transport address 140.113.20.10:5060, which is designated in the *Via* header field of the received SIP message (part B of the packets in Figure 2.2 (6), (7), and (8)). Upon receipt of the packet, the VPN server encapsulates it by the IP and the PPP and sends the tunneled packet, of which the source and the destination IP addresses are 140.113.20.1 and 140.113.10.1, through the VPN tunnel. At the NAT, the destination IP address is translated to 192.168.0.10, and the tunneled packet is then sent to UA1. UA1 decapsulates the packet to obtain the SIP message. Similarly, the RTP packets between UA1 and UA2 can be correctly delivered through the VPN tunnel.

2.2 The Static Route Solution

In Static Route [2], the application-layer transport address translation is performed at a UA in the private network, and a standard NAT is used to translate the transport address at the network and the transport layers. This solution requires that SIP/RTP related private-to-public transport address mappings are manually configured in both the UA and the NAT before a SIP call is set up. Specifically, both the UA and the NAT need to configure a *SIP mapping* (e.g., entry 1 in Table 2.1) and an *RTP mapping* (e.g., entry 2 in Table 2.1). If the UA is engaged in multiple media streams (e.g., audio plus video), extra RTP mappings are required.

Figure 2.3 illustrates SIP message delivery between UA1 (in the private network) and UA2 (in the Internet) based on the Static Route solution. The IP address settings for UA1, UA2, and the NAT are the same as those in Figure 2.1. Initially, the SIP mapping and the RTP mapping are configured in both UA1 and the NAT. Note that UA1 is modified to implement the mapping table while UA2 is a standard UA.

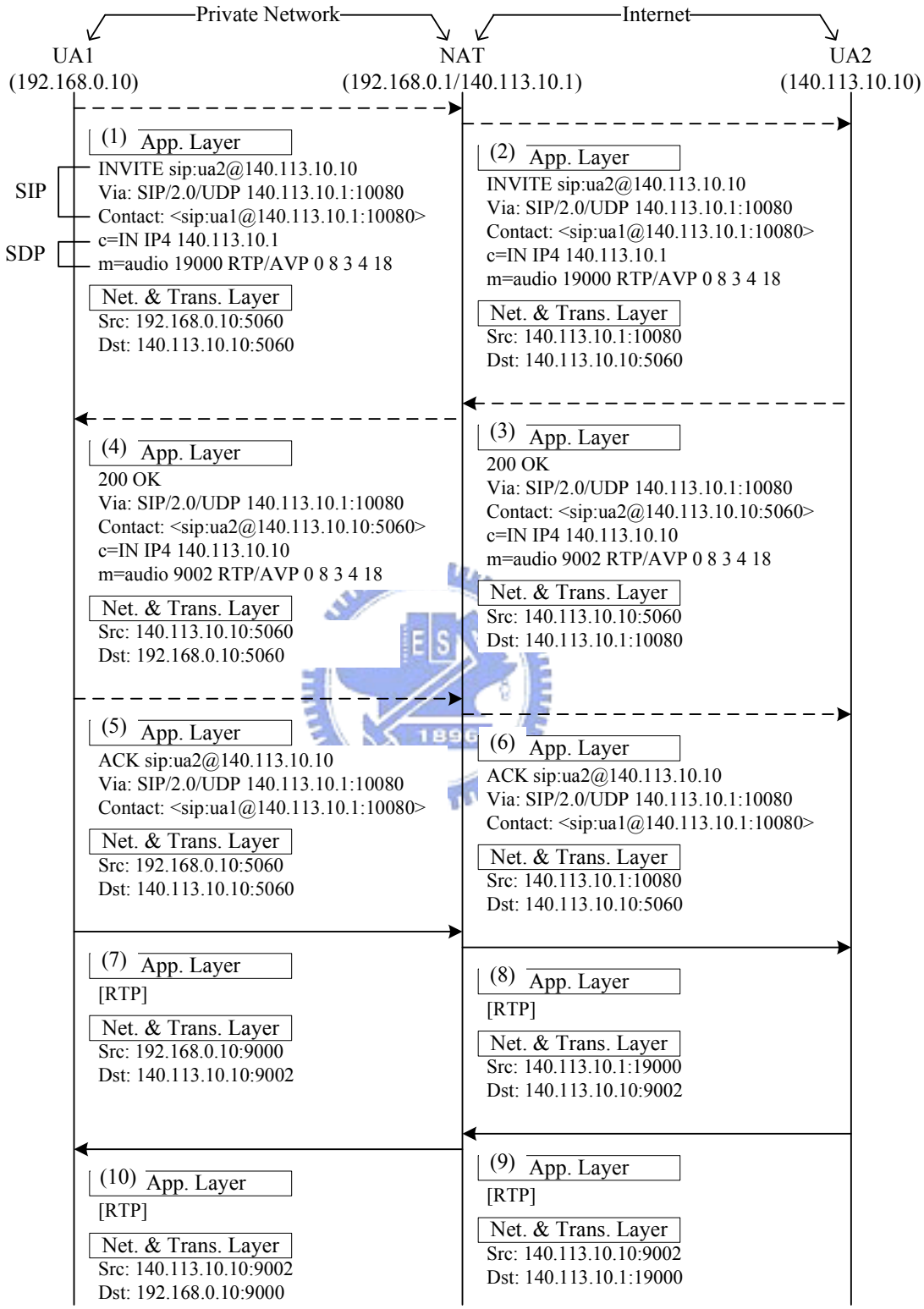


Figure 2.3: SIP/RTP NAT Traversing: the Static Route Solution

When UA1 sends a SIP INVITE message to UA2 (Figure 2.3 (1)), the message is carried by an IP packet with the source transport address 192.168.0.10:5060. The transport address reserved for the media session is 192.168.0.10:9000. The above private transport addresses are not shown in the application-layer content. Instead, through the mapping table in UA1, the private transport address 192.168.0.10:5060 is replaced by the mapped (public) transport address 140.113.10.1:10080, which is filled in both the *Via* and the *Contact* header fields. Also, the public IP address 140.113.10.1 and the public port number 19000 for RTP are filled in the *c* and the *m* fields, respectively. At the NAT, the source transport address is translated from 192.168.0.10:5060 to 140.113.10.1:10080 (Figure 2.3 (2)). The application-layer content is left unchanged.

Upon receipt of the INVITE message, UA2 replies a SIP 200 OK message (Figure 2.3 (3)). The *Via* header field (i.e., 140.113.10.1:10080) in the INVITE message (Figure 2.3 (2)) is copied to the 200 OK message as the destination of the message. Then the 200 OK message is sent to the NAT. Based on the mapping table, the NAT translates the destination transport address from 140.113.10.1:10080 to 192.168.0.10:5060 and sends the message to UA1 (Figure 2.3 (4)). After UA1 receives the 200 OK message, a SIP ACK message (with 140.113.10.1:10080 in the *Via* and the *Contact* header fields) is delivered to UA2 (Figure 2.3 (5) and (6)) just like the INVITE message.

The RTP packets from UA1 to UA2 are delivered to 140.113.10.10:9002 (Figure 2.3 (7)) as designated by the *c* and the *m* fields in the 200 OK message (Figure 2.3 (4)). At the NAT, the source transport address is translated from 192.168.0.10:9000 to 140.113.10.1:19000. These packets are then sent to UA2 (Figure 2.3 (8)). For the RTP packets sent from UA2 to UA1 (Figure 2.3 (9)), they are delivered to 140.113.10.1:19000, which is specified in the *c* and the *m* fields in the INVITE message (Figure 2.3 (2)). Upon receipt of the RTP packets, the NAT translates the destination transport address from 140.113.10.1:19000 to 192.168.0.10:9000 and sends the packets to UA1 (Figure 2.3 (10)).

2.3 The UPnP Solution

Manual configuration in the Static Route solution can be automated by UPnP [4]. UPnP is a network protocol for automatic discovery and configuration when a certain device (i.e., a UPnP client) is online. The solution for traversing NAT using UPnP is investigated by the UPnP Internet Gateway Working Committee [5]. Similar to Static Route, the SIP and the RTP mappings are configured in both the UA and the NAT before a SIP call is set up. Unlike Static Route, the standard UA is modified to support UPnP and to implement the mapping table. Also, the NAT must be modified to support UPnP because the mappings in both the UA and the NAT are automatically established by the UPnP protocol. After the mappings are confirmed, all SIP/RTP packets traverse over the NAT with the same procedure described in Section 2.2.

A UPnP system typically consists of several UPnP clients and one Internet Gateway Device (IGD). The IGD joins in the multicast group 239.255.255.250 [5] and listens on the port number 1900 for requests issued by the UPnP clients. The UPnP messages are exchanged through the Hypertext Transfer Protocol (HTTP) [14]. Figure 2.4 illustrates how the mapping in entry 1 of Table 2.1 is established by the UPnP messages exchanged between UA1 (a UPnP client) and the NAT (the IGD). The IP address settings for UA1 and the NAT are the same as those in Figure 2.1. The message flow in Figure 2.4 is described as follows.

Step 1. When UA1 is online, it sends a UPnP multicast M-SEARCH request (with the destination transport address 239.255.255.250:1900) to find the NAT (i.e., the IGD). M-SEARCH is a method defined by Simple Service Discovery Protocol (SSDP) [15] for service discovery.

Step 2. Upon receipt of the M-SEARCH request, the NAT returns its private location to UA1 by filling the transport address 192.168.0.1:2869 in the payload of a unicast HTTP response (i.e., 200 OK). The transport address is then used as the destination of the messages sent from UA1 to the NAT.

Step 3. To retrieve the mapped IP address, UA1 sends a UPnP GetExternalIPAddress

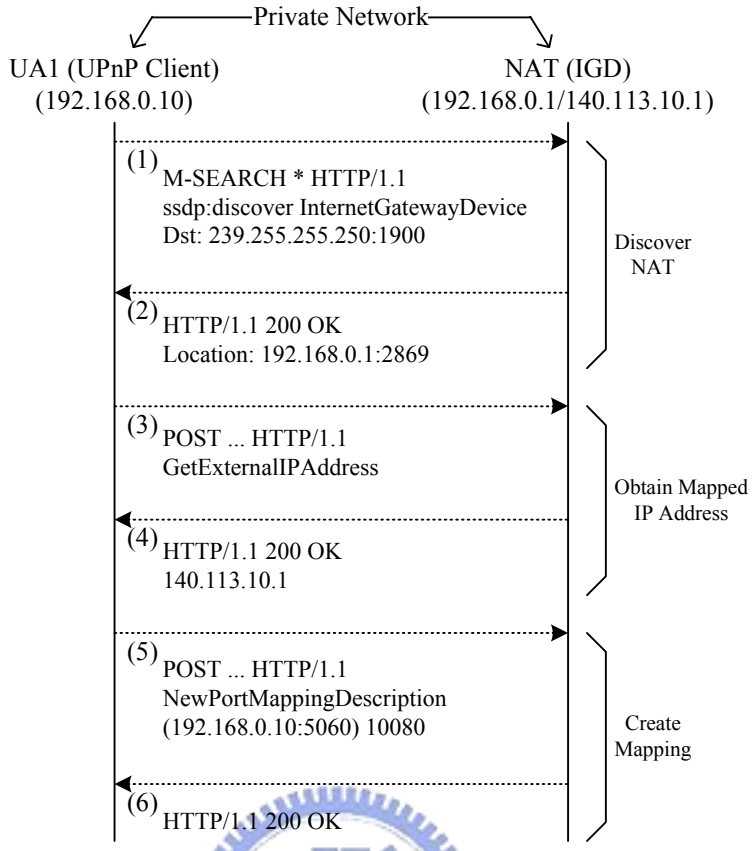


Figure 2.4: Establishing Private-to-public Transport Address Mapping through UPnP

request (an HTTP POST message) to the NAT.

Step 4. The NAT then replies its public IP address (i.e., 140.113.10.1) to UA1 through an HTTP 200 OK message.

Step 5. UA1 sends a UPnP NewPortMappingDescription request (an HTTP POST message) with a transport address and a port number "(192.168.0.10:5060) 10080", indicating the private transport address and the mapped port number in entry 1 of Table 2.1, respectively.

Step 6. If the proposed public mapped port number (i.e., 10080) is unused, the NAT confirms the mapping by replying an HTTP 200 OK message, and the procedure is completed.

In Steps 3 and 4, the NAT informs UA1 of the mapped IP address (i.e., the NAT's public IP address 140.113.10.1) for the corresponding private IP address. In Steps 5 and 6, UA1

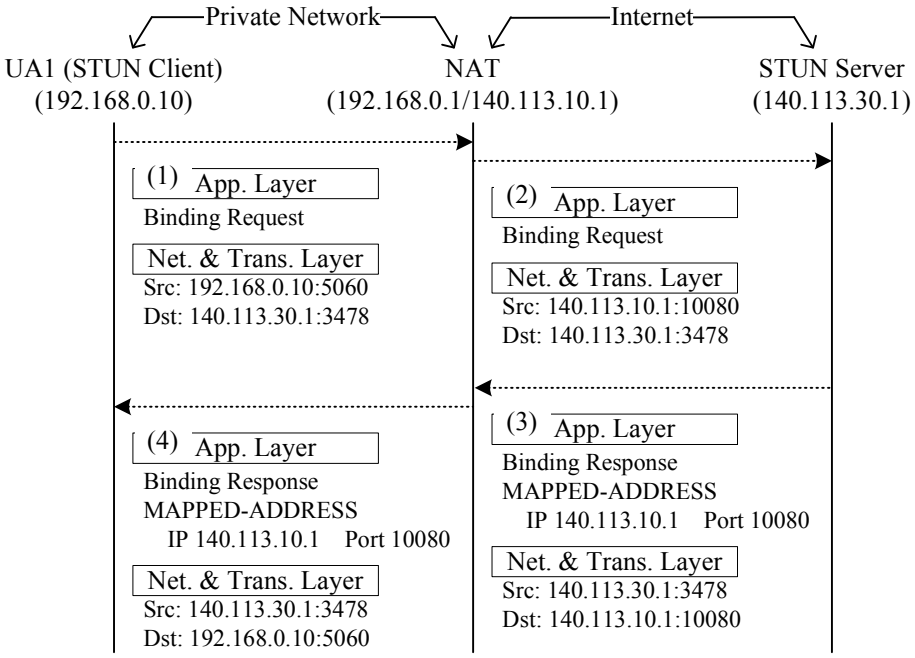


Figure 2.5: Establishing Private-to-public Transport Address Mapping through STUN

selects a port number for the mapped IP address and informs the NAT of the private-to-public transport address mapping. Steps 5 and 6 allow the applications (e.g., SIP or FTP) to specify well known port numbers (e.g., 5060 or 21) at the NAT. The UPnP solution also supports the deletion of mappings. This feature enables applications to create short-lived mappings for short session-based communications.

2.4 The STUN Solution

Like UPnP, STUN [20] supports automatic configuration of the SIP and the RTP mappings in both the UA and the NAT before a SIP call is set up. Also the standard UA is modified to support STUN and to implement the mapping table. Unlike UPnP, the standard NAT does not need any modification. Instead, an extra STUN server is required for the UA to obtain the mappings. After the mappings are confirmed, all SIP/RTP packets traverse over the NAT with the same procedure described in Section 2.2. At this stage, the STUN server needs not to involve.

A STUN system is typically composed of a STUN server in the Internet and several STUN clients in the private network. The STUN server listens on the port number 3478 for requests. Figure 2.5 illustrates how the mapping in entry 1 of Table 2.1 is established.

STUN messages in this figure are exchanged between UA1 (a STUN client) and the STUN server (with public IP address 140.113.30.1). The IP address settings for UA1 and the NAT are the same as those in Figure 2.1. The steps are described as follows.

Step 1. UA1 sends a STUN Binding Request message to the STUN server. This message is carried by an IP packet with the source transport address 192.168.0.10:5060.

Step 2. At the NAT, the mapping in entry 1 of Table 2.1 is established. According to the mapping table, the source transport address is translated from 192.168.0.10:5060 to 140.113.10.1:10080.

Step 3. Upon receipt of the message, the STUN server retrieves the source IP address 140.113.10.1 and the source port number 10080, and it then fills them separately in the *IP* and the *Port* fields of the *MAPPED-ADDRESS* attribute in a STUN Binding Response message. This Binding Response message is then carried by an IP packet with the destination transport address 140.113.10.1:10080 and sent to the NAT.

Step 4. The NAT retrieves the mapping from its mapping table, translates the destination transport address from 140.113.10.1:10080 to 192.168.0.10:5060, and sends the message to UA1. UA1 retrieves the transport address 140.113.10.1:10080 from the message and creates the mapping (i.e., entry 1 of Table 2.1) in its mapping table.

The NAT may eliminate the entries in its mapping table due to timeout. Therefore, the UA in the private network should periodically transmit the STUN Binding Request messages to refresh the mappings.

STUN does not support SIP/RTP NAT traversal over symmetric NAT. In a symmetric NAT, the private-to-public transport address mapping is affected by both the source and the destination transport addresses. In other words, two packets (sent from the private network to the Internet) with the same private source transport address but different destination transport addresses would be translated to those with the same public source IP address but different source port numbers. In this case, the mappings created through the STUN Binding Request/Response messages (stored in the UA's mapping table) are

different from those for the SIP/RTP packet delivery (stored in the NAT). Hence, the packets from the Internet cannot be delivered correctly to the UA in the private network.

2.5 The SIP-ALG Solution

A SIP-ALG [11] for SIP/RTP NAT traversal typically collocates with an NAT to create the SIP and the RTP mappings. The NAT and the SIP-ALG use these mappings to translate the SIP messages and the RTP packets. Specifically, when the NAT receives a packet, it identifies the packet as a SIP message by the source/destination port numbers (i.e., 5060) or the ASCII keyword "SIP/2.0" in the payload of the packet [12]. The SIP message is forwarded to the SIP-ALG for translation. Then the translated SIP message is returned to the NAT, and the NAT sends the SIP message to the destination.

Figure 2.6 illustrates SIP message delivery between UA1 (in the private network) and UA2 (in the Internet) based on the SIP-ALG solution. The IP address settings for UA1, UA2, and the NAT are the same as those in Figure 2.1.

Suppose that UA1 sends UA2 a SIP INVITE message with the source transport address 192.168.0.10:5060 (Figure 2.6 (1)). Upon receipt of the INVITE message, the NAT invokes the SIP-ALG to translate the message. Specifically, the SIP-ALG instructs the NAT to create a SIP mapping (i.e., entry 1 of the mapping table in Figure 2.6) and an RTP mapping (i.e., entry 2 of the mapping table in Figure 2.6). Based on the SIP mapping, the private transport address in the *Via* and the *Contact* header fields is translated from 192.168.0.10:5060 into 140.113.10.1:10080. Also, based on the RTP mapping, the private transport address in the *c* and the *m* fields is translated from 192.168.0.10:9000 into 140.113.10.1:19000. The modified INVITE message is then returned to the NAT. After the NAT translates the source transport address into 140.113.10.1:10080, the INVITE message is sent to UA2 (Figure 2.6 (2)).

Upon receipt of the INVITE message, UA2 replies a SIP 200 OK message (Figure 2.6 (3)). The *Via* header field (i.e., 140.113.10.1:10080) in the INVITE message is copied to the 200 OK message as the destination of the message. The 200 OK message is then sent to the NAT. At the NAT, the SIP-ALG is invoked to translate the *Via* header field from

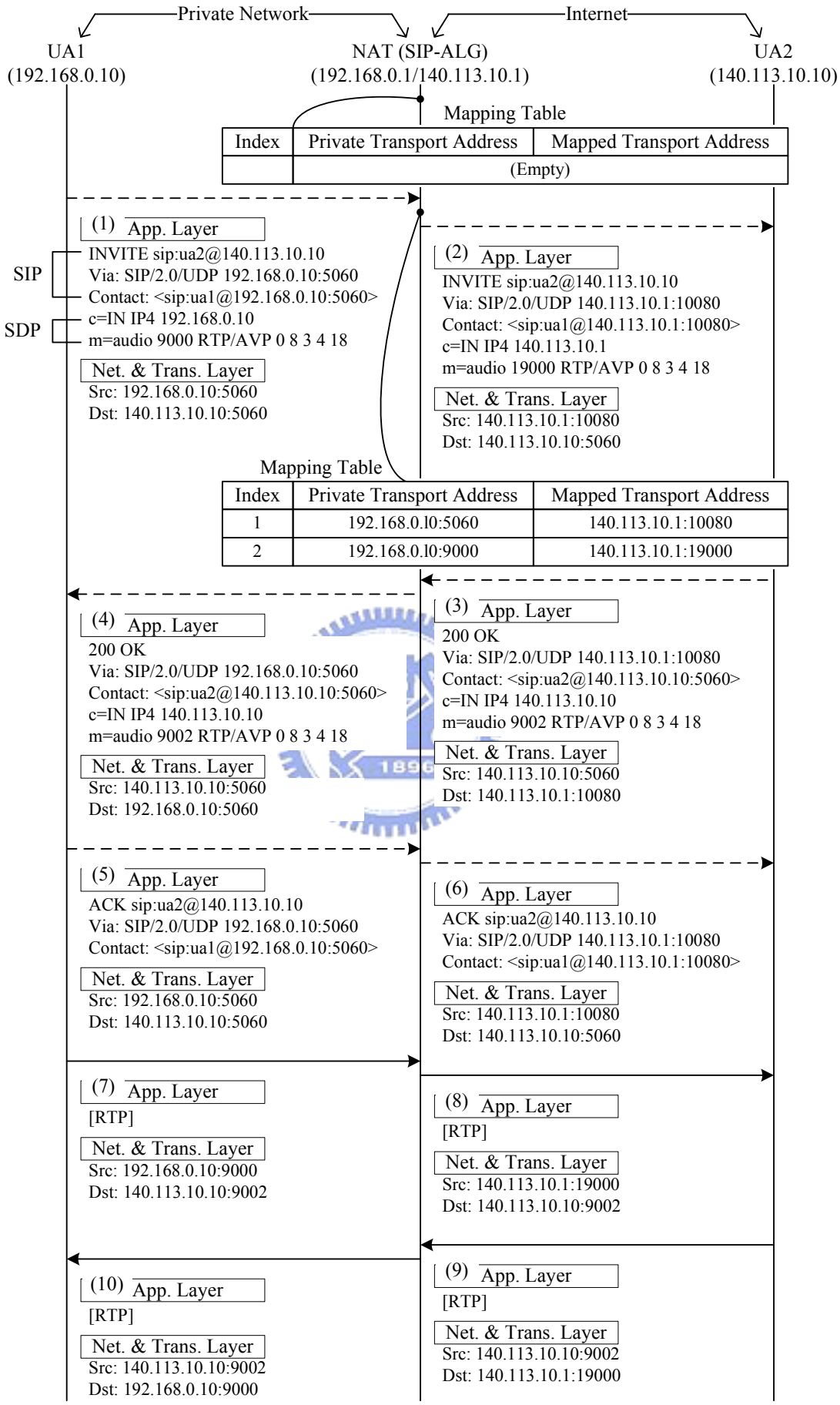


Figure 2.6: SIP/RTP NAT Traversing: the SIP-ALG Solution

140.113.10.1:10080 into 192.168.0.10:5060 based on the SIP mapping while the transport addresses in the *Contact* header field and the c/m fields need not be modified. The NAT then translates the destination transport address to 192.168.0.10:5060 and sends the 200 OK message to UA1 (Figure 2.6 (4)). Similar translation is applied to the SIP ACK message sent from UA1 to UA2 except that the ACK message does not contain the SDP description (see Figure 2.6 (5) and (6)).

The RTP packets from UA1 to UA2 are delivered to 140.113.10.10:9002 (Figure 2.6 (7)) as designated by the c and the m fields in the 200 OK message. Based on the RTP mapping, the NAT translates the source transport address into 140.113.10.1:19000 and sends these packets to UA2 (Figure 2.6 (8)). For the RTP packets from UA2 to UA1 (Figure 2.6 (9)), they are delivered to 140.113.10.1:19000, which is specified in the c and the m fields in the INVITE message received by UA2. After the NAT translates the destination transport address into 192.168.0.10:9000 based on the RTP mapping, the packets are sent to UA1 (Figure 2.6 (10)).

2.6 The SBC Solution

An SBC [9], locating in the Internet, serves as an outbound SIP proxy of the UAs in the private network so that all the SIP messages from the private network arrive at the SBC first. The SBC utilizes the *received* and the *rport* parameters to solve the SIP NAT traversing issue. The RTP NAT traversing issue is resolved by breaking one RTP media session into two. That is, instead of one RTP media session with the RTP packets directly delivered between the two UAs, two RTP media sub-sessions are set up: one from the UA in the private network to the SBC and the other from the SBC to the UA in the Internet. The details are elaborated in the following example as shown in Figure 2.7.

In Figure 2.7, UA1 (in the private network) attempts to establish a call to UA2 (in the Internet) and sets an SBC, which is assigned an IP address 140.113.40.1, as its outbound SIP proxy before any SIP message is issued. The IP address settings for UA1, UA2, and the NAT are the same as those in Figure 2.1.

Suppose that UA1 sends UA2 a SIP INVITE message with the source transport ad-

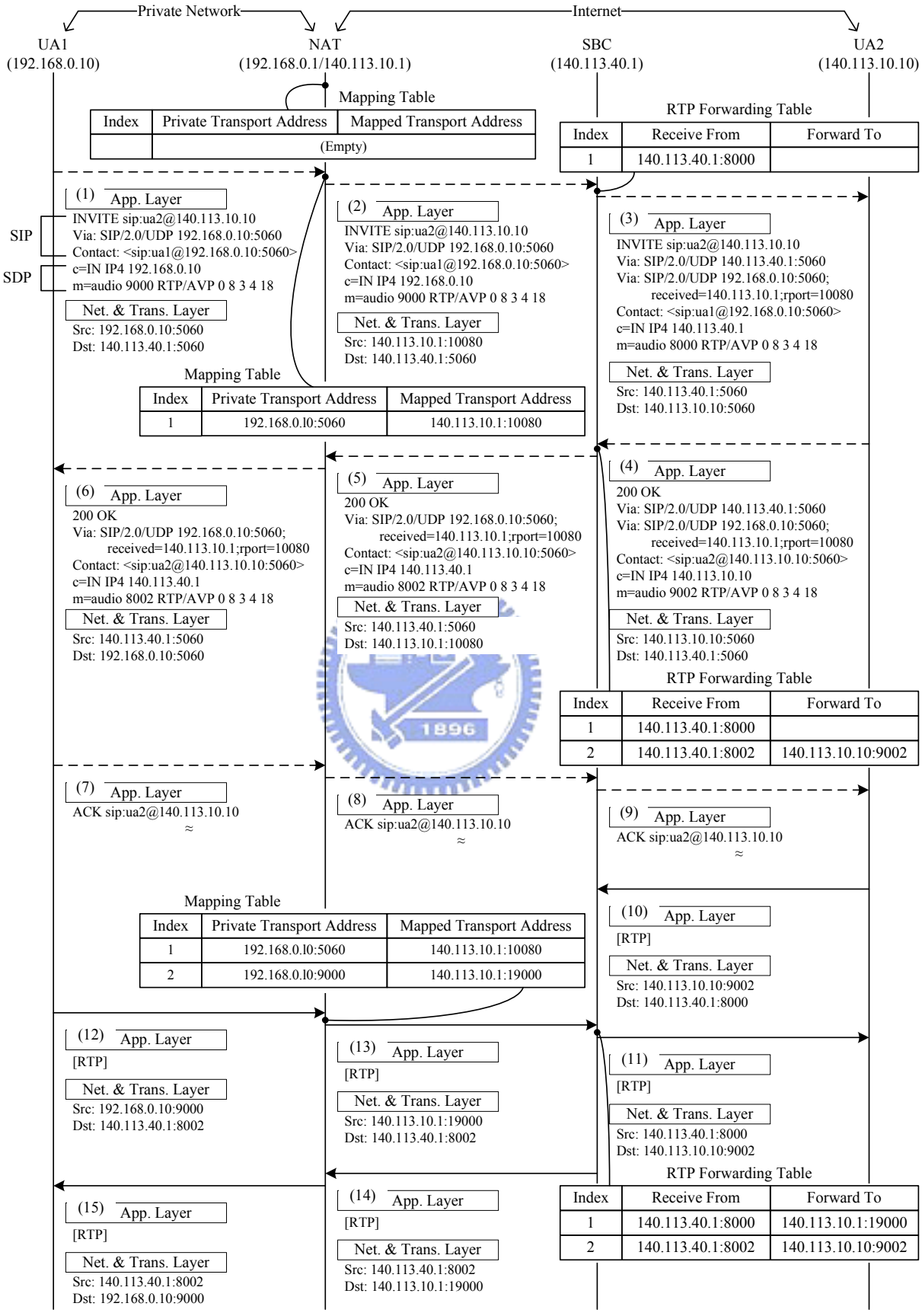


Figure 2.7: SIP/RTP NAT Traversing: the SBC Solution

dress 192.168.0.10:5060 (Figure 2.7 (1)). Upon receipt of the INVITE message, the NAT creates a SIP mapping (i.e., entry 1 of the mapping table in Figure 2.7), translates the source transport address into 140.113.10.1:10080, and sends the INVITE message to the SBC (Figure 2.7 (2)). When the SBC receives the INVITE message, it detects that the source transport address (i.e., 140.113.10.1:10080) is different from that in the *Via* header field (i.e., 192.168.0.10:5060). Hence, the SBC adds a *received* parameter with the IP address 140.113.10.1 and an *rport* parameter [22] with the port number 10080 to the *Via* header field for subsequent SIP response message delivery. In addition, to break the RTP media session into the UA1-SBC and the SBC-UA2 sub-sessions, the SBC replaces the private transport address 192.168.0.10:9000 in the *c* and the *m* fields by the SBC's transport address 140.113.40.1:8000 (i.e., the port number 8000 is used in the SBC for the SBC-UA2 sub-session). Then the SBC adds another *Via* header field with its transport address for SIP (i.e., 140.113.40.1:5060) and sends the INVITE message to UA2 (Figure 2.7 (3)).

Upon receipt of the INVITE message, UA2 replies a SIP 200 OK message with the *Via* header fields copied from the INVITE message (Figure 2.7 (4)). The 200 OK message is sent to the SBC according to the topmost *Via* header field (i.e., 140.113.40.1:5060). The SBC replaces the transport address in the *c* and the *m* fields with 140.113.40.1:8002 (i.e., the port number 8002 is used in the SBC for the UA1-SBC sub-session), removes the topmost *Via* header field, and sends the 200 OK message to UA1 according to the *received* and the *rport* parameters in the topmost *Via* header field (Figure 2.7 (5)). At the NAT, the destination transport address is translated into 192.168.0.10:5060, and the 200 OK message is sent to UA1 (Figure 2.7 (6)). The SIP ACK message from UA1 to UA2 is handled as the INVITE message except that the ACK message does not contain the SDP description (see Figure 2.7 (7), (8), and (9)). After UA2 receives the ACK message, the call is established.

For the SBC-UA2 sub-session, the RTP packets are delivered between the SBC and UA2 directly (Figure 2.7 (10) and (11)). For the UA1-SBC sub-session, UA1 learns from the 200 OK message the destination transport address for RTP (i.e., 140.113.40.1:8002) and sends the RTP packets to the SBC through the NAT (Figure 2.7 (12)). When these

RTP packets pass through the NAT, an RTP mapping is created (i.e., entry 2 of the mapping table in Figure 2.7) and used to translate the source transport address into 140.113.10.1:19000 (Figure 2.7 (13)). At the SBC, the source transport address of the RTP packets from UA1 is used as the destination transport address of the RTP packets to UA1 (Figure 2.7 (14)). In this way, the RTP packets from UA2 can arrive at UA1 correctly based on the RTP mapping (Figure 2.7 (15)).

2.7 The ICE Solution

ICE [19] is used to select the appropriate source/destination transport addresses for RTP packet delivery. The candidates of the transport addresses for RTP include the UA's local transport addresses and the mapped transport addresses obtained using the solutions described in the previous sections (e.g., the STUN solution). These candidates are exchanged during call setup by being filled in the *a=candidate* fields in the SDP description. In addition, the SIP NAT traversing issue is out of the scope of the ICE solution. Any solution can be adopted to cooperate with ICE, such as the usage of the *rport* parameter [22] described in Section 2.6. In ICE, both the UAs engaged in the RTP media session are required to support ICE so that the connectivity checks (which check the connectivity of the combination of a source and a destination transport addresses) can be performed. The connectivity checks are for selecting the most suitable source/destination transport addresses for RTP packet delivery. The details of the ICE solution are elaborated in the following example as shown in Figures 2.8 and 2.9.

Assume that UA1 (in the private network) obtains an RTP mapping through STUN and attempts to establish a call to UA2 (in the Internet). The IP address settings for UA1, UA2, the NAT, and the STUN server are the same as those in Figures 2.1 and 2.5. In this example, the symmetric NAT is adopted to show how ICE supports RTP NAT traversal over symmetric NAT while the mapped transport address for RTP is obtained through STUN (which does not support symmetric NAT traversal). The RTP mapping in the NAT's mapping table, which is created when UA1 obtains the RTP mapping through STUN, is shown in entry 1 of the mapping table in Figure 2.8.

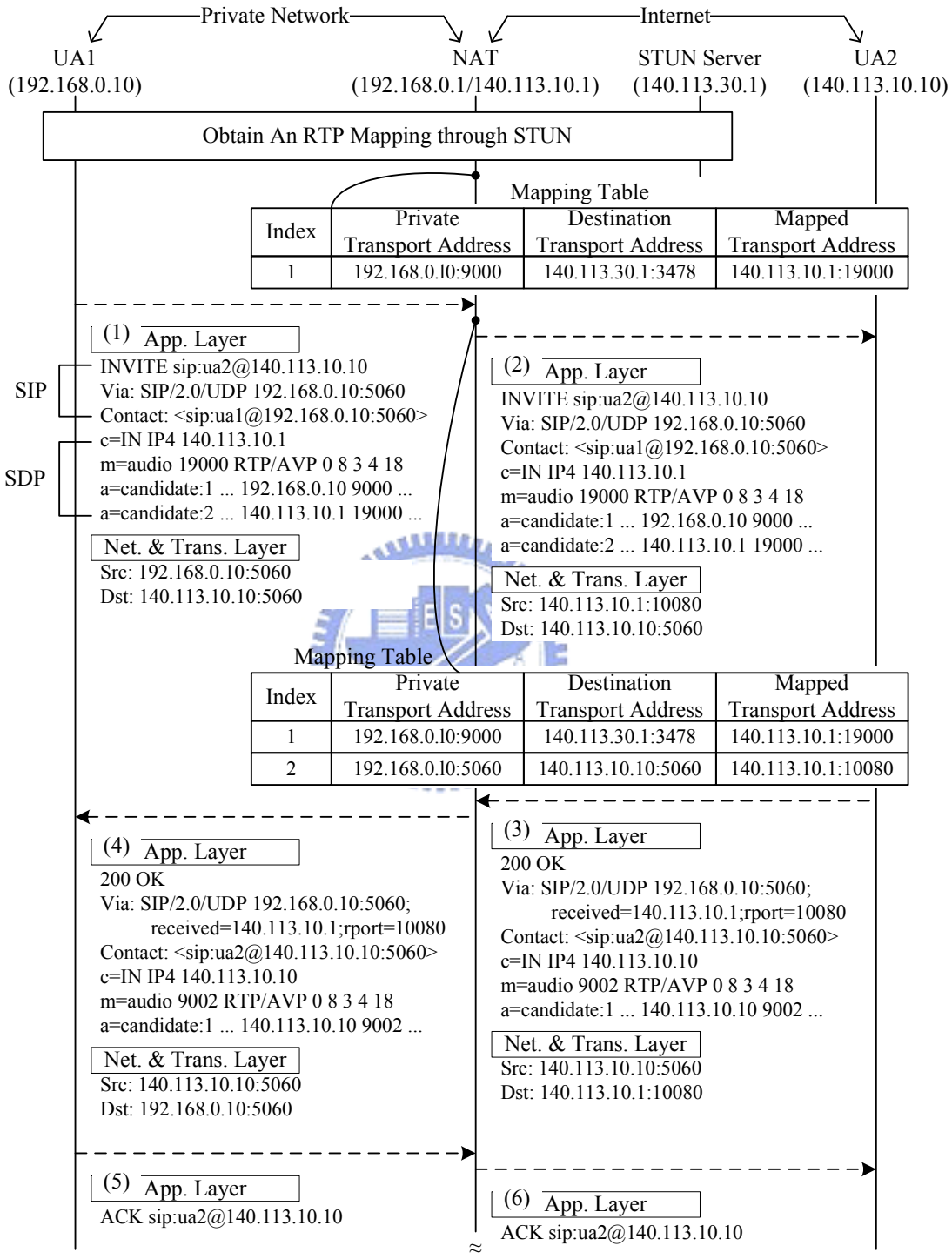


Figure 2.8: SIP/RTP NAT Traversing: Call Setup in the ICE Solution

Figure 2.8 illustrates the SIP message flow for call setup. The SIP NAT traversing issue is resolved by the usage of the *rport* parameter. The call setup procedure is similar to that in Section 2.6. During SIP message exchange, UA1 includes its private transport address 192.168.0.10:9000 and the corresponding mapped transport address 140.113.10.1:19000 in the *a=candidate* fields of the INVITE message (Figure 2.8 (1)). Similarly, UA2 includes its transport address 140.113.10.10:9002 in the *a=candidate* field of the 200 OK message (Figure 2.8 (3)). After exchange, UA1 combines its private transport address 192.168.0.10:9000 with UA2's transport address 140.113.10.10:9002 as a pair of the source/destination transport addresses. The pair is added to UA1's check list (i.e., entry 1 of UA1's check list in Figure 2.9). In addition, the pair with the source/destination transport addresses 140.113.0.1:19000/140.113.10.10:9002 is added to UA1's check list (i.e., entry 2 of UA1's check list in Figure 2.9). Since 140.113.10.1:19000 is the corresponding mapped transport address of 192.168.0.1:9000, the second entry is marked as associated to the first entry. That is, UA1 does not perform the connectivity check to the second entry because the transport address 140.113.10.1:19000 does not belong to UA1. (The transport address 140.113.10.1:19000 belongs to the NAT.) UA2 performs similar procedure to construct its check list (see UA2's check list in Figure 2.9) except that the second entry is not associated to the first entry.

Before sending the RTP packets to each other, UA1 and UA2 perform the connectivity checks to all the entries in their check lists, as illustrated in Figure 2.9. Suppose that UA1 checks the first entry in its check list. First, UA1 sends a STUN Binding Request message with the source transport address 192.168.0.10:9000 and the destination transport address 140.113.10.10:9002 (Figure 2.9 (1)). Upon receipt of the message, the NAT creates a new private-to-public transport address mapping (i.e., entry 3 of the NAT's mappings table in Figure 2.9), translates the source transport address into 140.113.10.1:29000, and forwards the message to UA2 (Figure 2.9 (2)). Note that the source port number of the message received by UA2 (i.e., 29000) is different from UA1's mapped port number specified in the INVITE message (i.e., 19000). Therefore, UA2 adds to its check list a new pair with the source/destination transport addresses 140.113.10.10:9002/140.113.10.1:29000 (i.e., entry

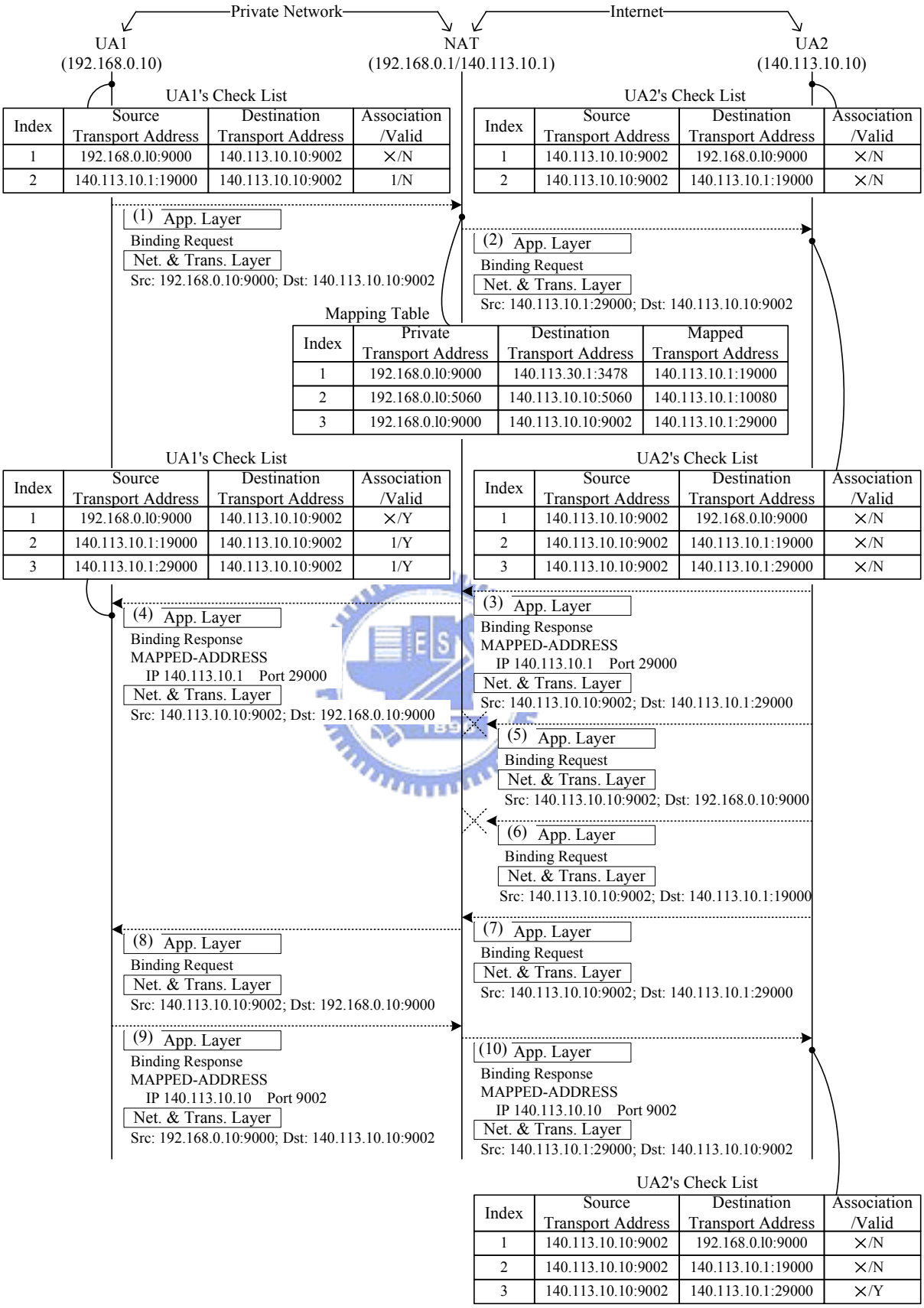


Figure 2.9: SIP/RTP NAT Traversing: Connectivity Checks in the ICE Solution

3 of UA2's check list in Figure 2.9). UA2 then replies a STUN Binding Response message to UA1 (Figure 2.9 (3)). When UA1 receives this message, the first entry in its check list is marked "valid" (Figure 2.9 (4)). Besides, UA1 learns from the message a new mapped transport address (i.e., 140.113.10.1:29000). UA1 then adds to its check list a new pair with the source/destination transport addresses 140.113.10.1:29000/140.113.10.10:9002 (i.e., entry 3 of UA1's check list in Figure 2.9). Like the second entry, the new entry is marked as associated to the first entry, and no connectivity check is performed to it.

UA2 performs similar procedure to check all the entries in its check list. For the first entry, the destination transport address 192.168.0.10:9000 is unreachable from the Internet (Figure 2.9 (5)). For the second entry, there is no mapping in the NAT's mapping table that satisfies the condition where the destination transport address is 140.113.10.10:9002 and the mapped transport address is 140.113.10.1:19000 (Figure 2.9 (6)). For the third entry, the STUN Binding Request message from UA2 is sent to UA1 according to entry 3 in the NAT's mapping table in Figure 2.9 (Figure 2.9 (7) and (8)). After receiving the STUN Binding Response message from UA1 (Figure 2.9 (9) and (10)), UA2 marks the third entry "valid." After the connectivity checks, both UA1 and UA2 select the source/destination transport addresses for RTP from the entries that are marked "valid." In this example, UA1 selects 192.168.0.10:9000/140.113.10.10:9002, and UA2 selects 140.113.10.10:9002/140.113.10.1:29000. Therefore, the RTP packets delivered between UA1 and UA2 are translated by the NAT according to entry 3 in the NAT's mapping table in Figure 2.9.

Consider another example where UA1 and UA2 reside in the same private network. Both UA1 and UA2 obtain the mapped transport addresses through STUN and exchange their private and mapped transport addresses during call setup. After the connectivity checks, they detect that they can send the RTP packets to each other directly using each other's private transport address. This avoids the address translation performed at the NAT. Therefore, in this case, ICE provides more efficient RTP packet delivery than the other solutions.

Table 2.2: Comparisons of the SIP/RTP NAT Traversing Solutions

Solution	UA Modification	Scope of NATs Supported	Multi-layer NAT Traversal	Extra Server	Extra Public IP Address	Configuration Complexity
VPN	Yes	All Types of NATs	Yes	Yes	Yes	Medium
Static Route	Yes	All Types of NATs	Yes	No	No	High
UPnP	Yes	NATs Supporting UPnP	No	No	No	Low
STUN	Yes	All Types of NATs except Symmetric NAT	Yes	Yes	No	Medium
ICE	Yes	All Types of NATs	Yes	Yes	No	Low
SIP-ALG	No	NATs Supporting SIP-ALG	Yes	No	No	Low
SBC	No	All Types of NATs	Yes	Yes	No	Medium

2.8 Comparisons

This section compares the SIP/RTP NAT traversing solutions. The compared items are listed in Table 2.2 and described as follows.

UA Modification: For VPN, the UA is equipped with the VPN software to establish the VPN tunnel. For ICE, not only the UA in the private network but also the UA in the Internet are equipped with the ICE software. For SIP-ALG and SBC, the UA does not require any modification (except for outbound SIP proxy setting in SBC). For Static Route, UPnP, and STUN, the standard UA is modified to perform two tasks. In the first task, the SIP and the RTP mappings at the NAT are obtained through manual setting, UPnP, or STUN. In the second task, the UA translates the transport address in the SIP messages. For example, this task can be achieved by the functions in eXtended osip (eXosip) library [3]:

- The **eXosip_set_firewallip()** uses the specific IP address in the *Via* header field, the *Contact* header field, and the *c* field in the generated SIP messages (i.e., in which the mapped IP address in Table 2.1 should be filled).
- The **eXosip_initiate_call()** fills the indicated port number in the *m* field in the generated SIP messages.

The problem of **eXosip_set_firewallip()** is that it does not specify the port number

in both the *Via* and the *Contact* header fields. Instead, the source port number of those IP packets carrying the generated SIP messages is used. To resolve this issue, we have added a function **eXosip_set_firewallsipPort()** that allows the UA to specify the port number filled in both the *Via* and the *Contact* header fields.

Scope of NATs Supported: VPN, Static Route, ICE, and SBC support SIP/RTP traversal over all types of NATs. The NATs need not be modified. In UPnP and SIP-ALG, an agent should be collocated with the NAT to serve as an IGD and a SIP-ALG, respectively. Although no modification is made to the NAT for STUN, STUN does not support SIP/RTP traversal over symmetric NAT.

Multi-layer NAT Traversal: When a UA resides in a private network within another private network (therefore there are multi-layer NATs), all solutions but UPnP support SIP/RTP NAT traversal. UPnP does not work because a UPnP client can only identify the NAT closest to it. It does not have any knowledge to pass through the other NATs.

Extra Server: VPN, STUN, ICE, and SBC require extra servers in the Internet. Static Route does not require any extra server. For UPnP, the NAT is modified as a UPnP server. SIP-ALG typically resides in the NAT.

Extra Public IP Address: In VPN, an extra public IP address is required for each VPN tunnel. Other solutions do not require any extra public IP address.

Configuration Complexity: Since the SIP and the RTP mappings are automatically established by UPnP, ICE, and SIP-ALG, these solutions do not incur any configuration cost. In VPN and STUN, a user configures the server location and the user name/password for authentication. In SBC, the UA sets the SBC as the outbound SIP proxy. On the other hand, a Static Route user should manually configure the SIP and the RTP mappings for SIP/RTP in both the UA and the NAT.

Furthermore, We conduct measurements in an experimental environment (as shown in Figure 2.10) where the UA in the private network (UA1) connects to the NAT through

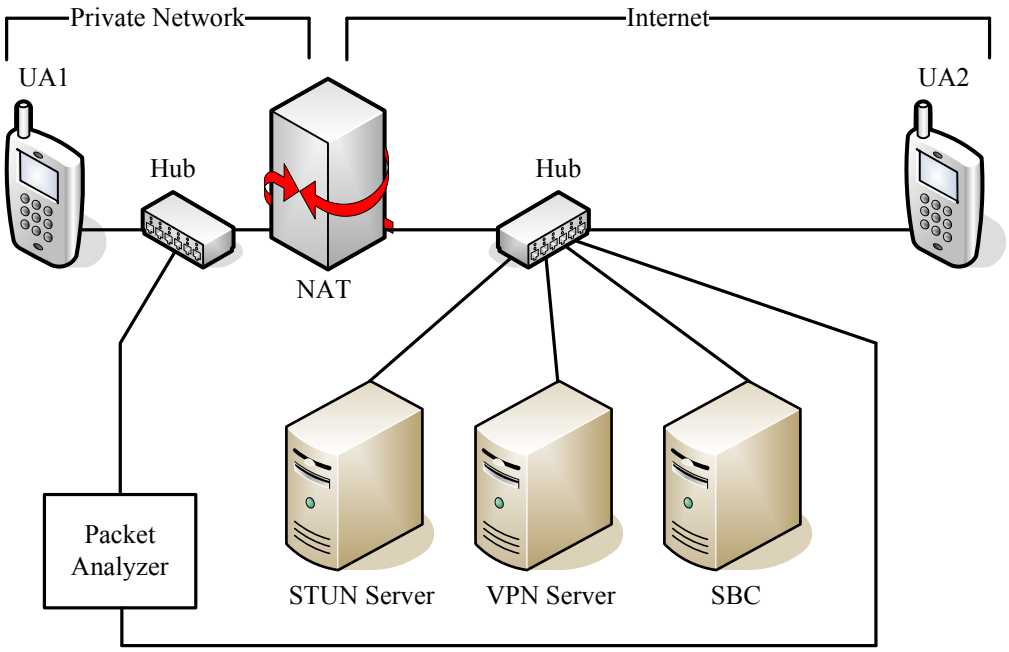


Figure 2.10: Experimental Environment for SIP/RTP NAT Traversing Solutions

a hub. The other UA (UA2), the VPN server, the STUN server, and the SBC reside in the Internet and connect to the NAT through another hub. To provide the same baseline for the following performance comparisons, a UA that supports all the SIP/RTP NAT traversing solutions discussed in this chapter is required. Therefore, we design and develop a UA using eXosip library [3] to which some modifications are made as described above.

Table 2.3 shows the time complexities of the solutions discussed in this chapter except ICE. Since ICE utilizes the solutions described in this chapter for resolving SIP NAT traversing issue and for obtaining the RTP mappings, the costs for private-to-public transport address mapping establishment, the call setup latency, and the RTP latency are the same as those of the solution utilized in ICE. Moreover, the connectivity checks increase the call setup latency for ICE.

The table indicates that UPnP has the longest private-to-public transport address mapping establishment time, and VPN requires long VPN connection establishment time. In terms of SIP setup and RTP delivery, Static Route, UPnP, and STUN are better than the server-based solutions, which are in turn better than VPN.

Table 2.3: Time Complexities of the SIP/RTP NAT Traversing Solutions

Solution	Private-to-public Transport Address Mapping Establishment	VPN Connection Establishment	Call Setup	RTP Latency
VPN	N/A	2114 ms	161 ms	8.3 ms
Static Route	Manual Setup	N/A	71 ms	0.7 ms
UPnP	261 ms	N/A	71 ms	0.7 ms
STUN	27 ms	N/A	71 ms	0.7 ms
SIP-ALG	N/A	N/A	96 ms	0.7 ms
SBC	N/A	N/A	96 ms	4.3 ms

2.9 Summary

In this chapter, five UA-based (VPN, Static Route, UPnP, STUN, and ICE) and two server-based (SIP-ALG and SBC) SIP/RTP NAT traversing solutions are investigated. Our study indicates that VPN's time complexity is much higher than those of the other solutions. Static Route requires manual setting, which is considered inefficient. UPnP automates Static Route. However, UPnP requires NAT modification and cannot traverse over multi-layer NATs. STUN also automates Static Route while it requires an extra server and cannot traverse symmetric NAT. ICE supports symmetric NAT traversal and automates the selection of the source/destination transport addresses for RTP through connectivity checks. Nevertheless, ICE requires both the UAs engaged in the RTP media session to support ICE, and the connectivity checks introduce extra call setup latency. Both SIP-ALG and SBC automate translation of SIP messages without any modification to UAs. The NAT needs to be modified to accommodate the SIP-ALG, and an extra server is required for SBC. The call setup time and the RTP latency for the server-based solutions are longer than those for the UA-based solutions (except for VPN). In summary, there is no SIP/RTP NAT traversing solution that is better than the others in all aspects.

Chapter 3

IPv4/IPv6 Interworking for SIP-based VoIP

In the early stage of IPv6 deployment, most newly developed UAs support both IPv4 and IPv6 (called *IPv4/IPv6 dual-stack UAs*). Without a proper mechanism, IPv4/IPv6 interworking problem may occur when IPv4/IPv6 dual-stack UAs attempt to interact with the UAs that support IPv4 only (*IPv4-only UAs*). For example, when an IPv4/IPv6 dual-stack UA initiates a call by sending an IPv6 SIP INVITE message to an IPv4-only UA, the call cannot be established correctly. The IPv4-only UA may ignore the IPv6 INVITE message because it is unable to process the SIP messages that contain IPv6 addresses. Furthermore, the IPv4-only UA cannot send the RTP packets to the IPv4/IPv6 dual-stack UA because the destination of the RTP packets resides in the IPv6 network. That is, IPv6 network is unreachable for an IPv4 host. To resolve this problem, three solutions have been proposed in the literature: the SIP-ALG solution for IPv4/IPv6 interworking [26] ("the SIP-ALG solution" for short in this chapter), the Redirect solution [17], and the ICE solution for IPv4/IPv6 interworking [19] ("the ICE solution" for short in this chapter). Since these existing solutions either degrade the RTP transmission performance or introduce extra call setup latency, we propose an effective solution where the call setup and the RTP transmission overheads are minimized. All the four solutions will be elaborated and analyzed in the following sections.

For the sake of discussion, the IPv4/IPv6 interworking environment for SIP-based VoIP is simplified as illustrated in Figure 3.1. The calling party (UA1; see Figure 3.1 (1)) is an IPv4/IPv6 dual-stack UA. The called party (UA2; see Figure 3.1 (2)) is an IPv4-

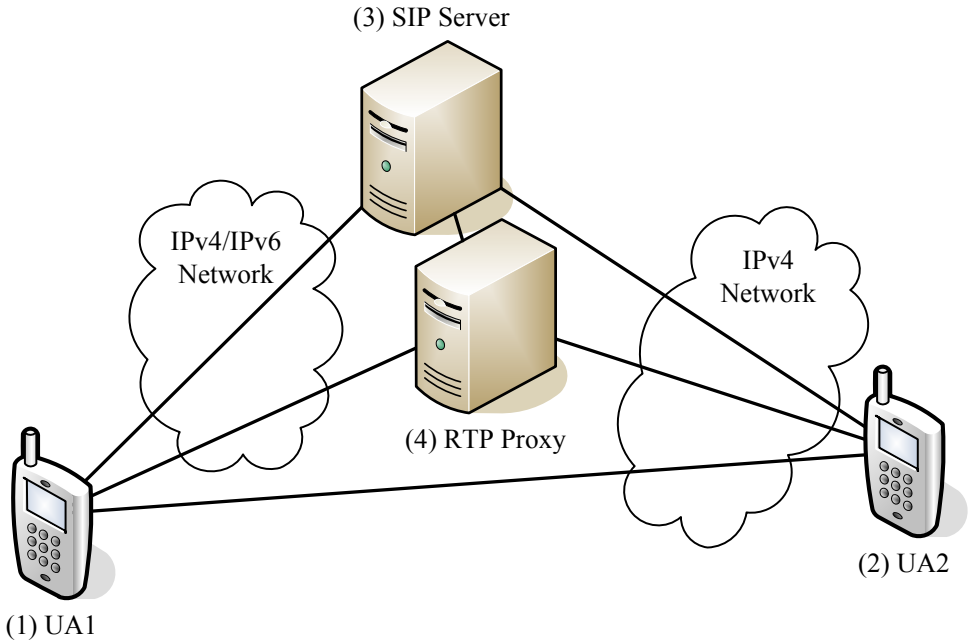


Figure 3.1: IPv4/IPv6 Interworking Environment for SIP-based VoIP

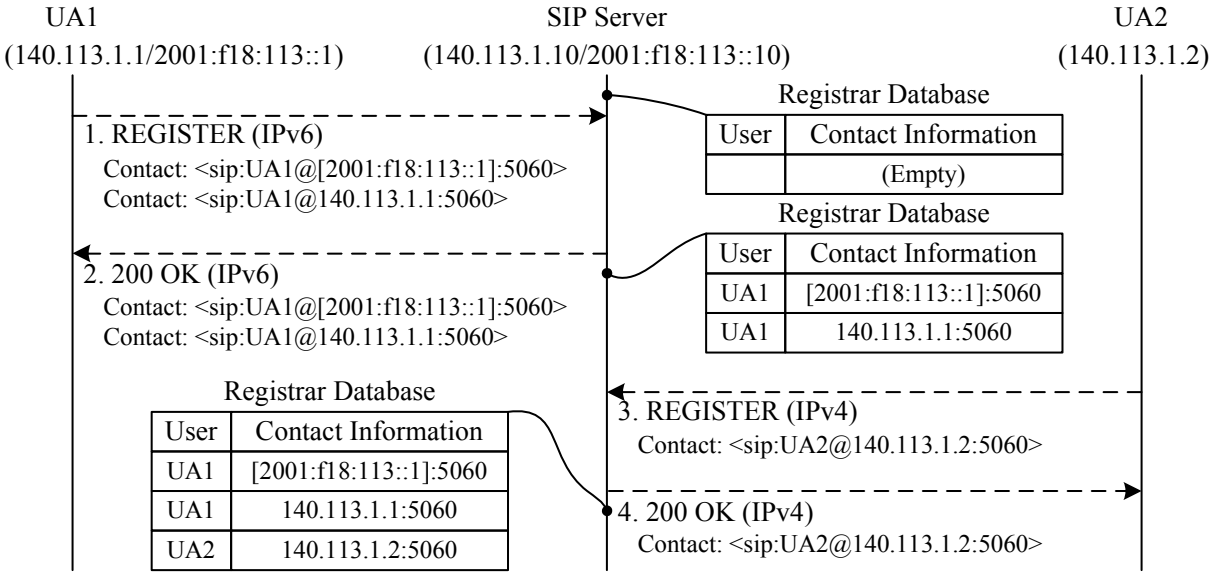


Figure 3.2: SIP Registration Message Flow

only UA. An IPv4/IPv6 dual-stack SIP server (Figure 3.1 (3)) functions as a registrar and proxy. An RTP proxy (Figure 3.1 (4)) is responsible for translating IPv4 RTP packets to IPv6 RTP packets, and vice versa. For the four solutions discussed in this chapter, the SIP server is required for all solutions while the RTP proxy is used in the SIP-ALG solution only. Without loss of generality, we assume that both UA1 and UA2 register their contact information at the SIP server in all the four solutions. Consider the SIP registration message flow in Figure 3.2 where UA1's IPv4/IPv6 addresses are 140.113.1.1 and 2001:f18:113::1. UA2's IPv4 address is 140.113.1.2. The SIP server's IPv4/IPv6 addresses are 140.113.1.10 and 2001:f18:113::10, respectively. In the registration procedure, UA1 first sends an IPv6 SIP REGISTER message to the SIP server with both its IPv6 and IPv4 contact information filled in two separate *Contact* header fields (Step 1 in Figure 3.2). The SIP server then stores the contact information in its registrar database and replies an IPv6 SIP 200 OK message indicating that the registration is successful (Step 2 in Figure 3.2). UA2 performs similar procedure to register its IPv4 contact information (Steps 3 and 4 in Figure 3.2). After registration, a UA can be reached through the SIP server. Furthermore, we assume that when UA1 initiates a call, it always attempts to send an IPv6 SIP INVITE message [10] [25].

3.1 The SIP-ALG Solution

In the SIP-ALG solution [26], the SIP server cooperates with the RTP proxy to translate the SIP/RTP packets delivered between UA1 and UA2. The call setup procedure is illustrated in Figure 3.3 with the following steps.

Step 1. UA1 sends an IPv6 SIP INVITE message to UA2 through the SIP server.

Step 2. Upon receipt of the INVITE message, the SIP server retrieves UA1's IP address from the *Contact* header field of the INVITE message and UA2's IP address from the SIP server's registrar database. The SIP server then compares the versions of UA1's and UA2's IP addresses. This action is called *IP version comparison*. In this example, the versions of UA1's and UA2's IP addresses are IPv6 and IPv4, respectively. To establish the call without changing the IP version that UA1 chooses

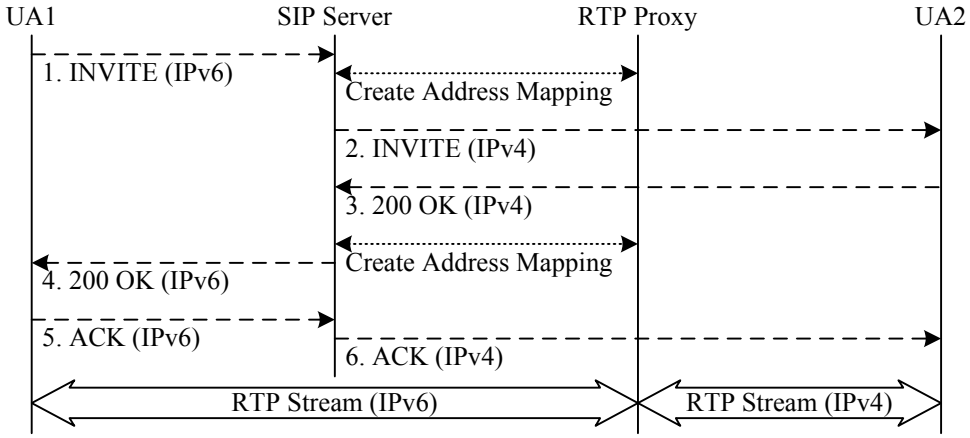


Figure 3.3: IPv4/IPv6 Interworking for SIP-based VoIP: the SIP-ALG Solution

for the call, the SIP server acts as a SIP-ALG to translate the SIP messages for this call setup. For the INVITE message, UA1's IPv6 transport address for SIP in the *Via* and the *Contact* header fields is replaced by the SIP server's IPv4 transport address. In addition, UA1's IPv6 transport address for RTP in the SDP description is passed to the RTP proxy to build an *IPv4-to-IPv6 transport address mapping* for forwarding the RTP packets from UA2 to UA1. The RTP proxy returns its IPv4 address and an unused port number to the SIP server for modifying the SDP description. After modification, the IPv4 INVITE message is then forwarded to UA2.

Step 3. Upon receipt of the INVITE message, UA2 retrieves the *Via* header field in the INVITE message. When UA2 accepts the call, an IPv4 SIP 200 OK message with the retrieved *Via* header field is created. The IPv4 200 OK message is then sent to the SIP server according to the *Via* header field.

Step 4. Upon receipt of the 200 OK message, the SIP server replaces its IPv4 transport address in the *Via* header field by UA1's IPv6 transport address for SIP. Also, UA2's IPv4 transport address for SIP in the *Contact* header field is replaced by the SIP server's IPv6 transport address. Similar to Step 2, the SIP server passes UA2's IPv4 transport address for RTP in the SDP description to the RTP proxy to build an *IPv6-to-IPv4 transport address mapping* for forwarding the RTP packets from UA1 to UA2. Then the SIP server modifies the SDP description using the RTP proxy's

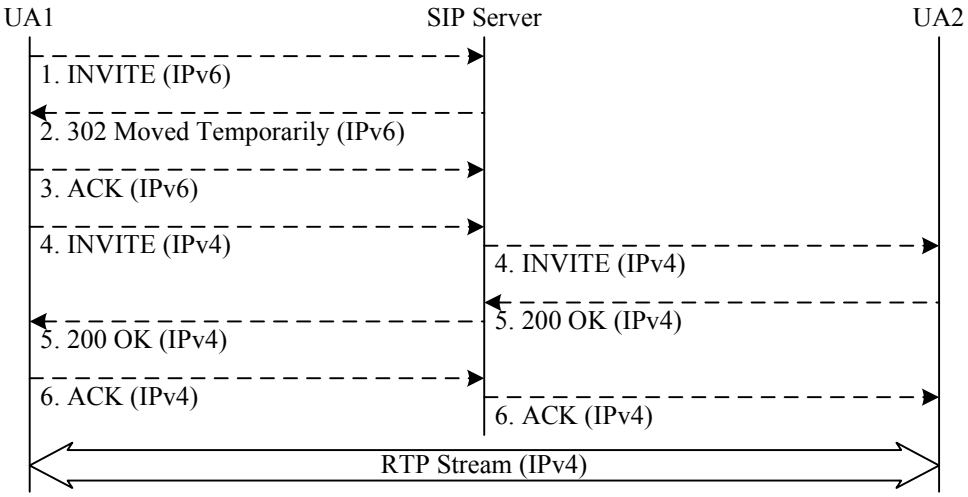


Figure 3.4: IPv4/IPv6 Interworking for SIP-based VoIP: the Redirect Solution

IPv6 transport address that is returned from the RTP proxy. According to the *Via* header field, the IPv6 200 OK message is forwarded to UA1.

Steps 5 and 6. When UA1 receives the 200 OK message, it replies an IPv6 SIP ACK message. The SIP server replaces UA1's IPv6 transport address for SIP in the *Via* and the *Contact* header fields by the SIP server's IPv4 transport address and forwards the IPv4 ACK message to UA2.

After the call is established, UA1 sends IPv6 RTP packets to UA2 through the RTP proxy. The RTP proxy translates these packets into IPv4 RTP packets and forwards them to UA2. Similarly, the IPv4 RTP packets from UA2 are translated into IPv6 RTP packets and then forwarded to UA1 by the RTP proxy.

3.2 The Redirect Solution

In the Redirect solution [17], upon receipt of the IPv6 SIP INVITE message from UA1, the SIP server informs UA1 to set up the call to UA2 through IPv4. The call setup procedure in Figure 3.4 is described as follows.

Step 1. UA1 sends an IPv6 SIP INVITE message to UA2 through the SIP server.

Step 2. Upon receipt of the INVITE message, the SIP server performs IP version comparison. Since the versions of UA1's and UA2's IP addresses are different, the SIP

server sends an IPv6 302 Moved Temporarily message to UA1 to indicate that IPv4 should be used for call setup.

Step 3. UA1 learns that UA2 is in the IPv4 domain and replies an IPv6 ACK message to the SIP server.

Step 4. UA1 sends an IPv4 SIP INVITE message to UA2 through the SIP server.

Steps 5 and 6. UA2 accepts the call and sends an IPv4 200 OK message to UA1 through the SIP server. UA1 then replies an IPv4 ACK message.

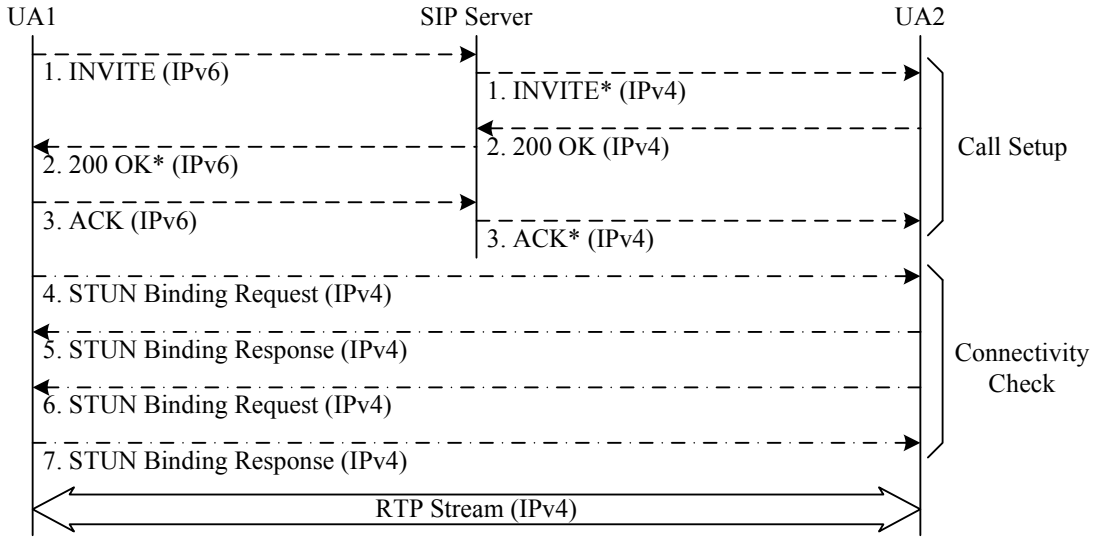
The call is established, and IPv4 RTP packets are delivered between UA1 and UA2.

3.3 The ICE Solution

In the ICE solution [19], UA1 and UA2 are responsible for selecting an appropriate IP version for RTP packet delivery. The SIP server translates the IPv4 packets (at the IP layer) that carry the SIP messages (at the application layer) to IPv6 packets, and vice versa. However, the transport addresses in these SIP messages are not modified by the SIP server except that the SIP server will add to the INVITE message an extra *Via* header field containing its IPv4 transport address for SIP (so that UA2 can route back the 200 OK message). The call setup procedure (see Figure 3.5) is described in the following steps.

Step 1. UA1 sends an IPv6 SIP INVITE message to UA2 through the SIP server. In addition to its IPv6 transport address for RTP, UA1 includes its IPv4 transport address for RTP in the SDP description. Upon receipt of the INVITE message, the SIP server retrieves UA2's contact information from its registrar database and learns that UA2 supports IPv4 only. The SIP server then encapsulates the INVITE message in an IPv4 packet and forwards it to UA2.

Step 2. When UA2 receives the INVITE message, the *Via* header fields are retrieved (to be used for creating a subsequent 200 OK message). In addition, UA1's IPv4 transport address for RTP in the SDP description is retrieved (to be used in Step 6).



* The IP packets that carry the INVITE and the ACK messages are changed to the IPv4 format. On the other hand, the transport addresses filled by UA1 in the carried messages are not changed by the SIP server. Similarly, the IP packet that carries the 200 OK message is changed to the IPv6 format, but UA2's IPv4 transport address in the 200 OK message is not changed.

Figure 3.5: IPv4/IPv6 Interworking for SIP-based VoIP: the ICE Solution

Since UA2 supports IPv4 only, other IPv6 transport addresses in the message are ignored. UA2 then creates an IPv4 200 OK message with the retrieved *Via* header fields and includes its IPv4 transport address for RTP in the SDP description. The 200 OK message is sent to the SIP server in an IPv4 packet. The SIP server then forwards the 200 OK message to UA1 in an IPv6 packet.

Step 3. After receiving the 200 OK message, UA1 retrieves UA2's IPv4 transport address for RTP from the SDP description (to be used in Step 4) and replies an IPv6 ACK message. Similar to the INVITE message, the ACK message is sent to the SIP server in an IPv6 packet and then forwarded to UA2 in an IPv4 packet by the SIP server.

Unlike the SIP-ALG and the Redirect solutions, the source/destination transport addresses for RTP packet delivery are not confirmed in the ICE-based call setup procedure (Steps 1–3). Instead, they are confirmed after the connectivity checks, which are performed before UA1 and UA2 send the RTP packets to each other. The connectivity checks utilize STUN Binding Request/Response messages. Suppose that UA1's IPv4 transport address for RTP is X and UA2's IPv4 transport address for RTP is Y. The

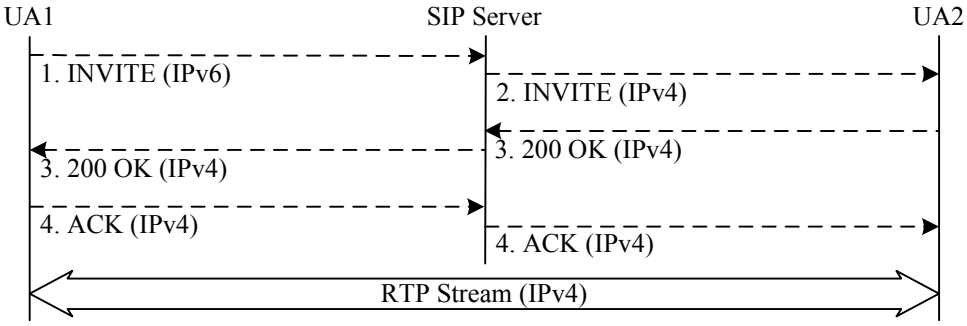


Figure 3.6: IPv4/IPv6 Interworking for SIP-based VoIP: Our Solution

connectivity check procedure (see Steps 4–7 in Figure 3.5) is described as follows.

Step 4. To confirm that Y can be used for RTP packet delivery, UA1 sends UA2 a STUN Binding Request message with the source transport address X and the destination transport address Y.

Step 5. Upon receipt of the Binding Request message, UA2 replies a STUN Binding Response message with the source transport address Y and the destination transport address X. When UA1 receives this message, it confirms that the RTP packets with the source transport address X and the destination transport address Y can be sent to UA2.

Steps 6 and 7. UA2 performs similar procedure to confirm that X can be used for RTP packet delivery.

After the connectivity checks, IPv4 RTP packets are delivered between UA1 and UA2.

3.4 Our Solution

In our solution, the SIP server translates the IPv6 SIP INVITE message based on the contact information stored in the SIP server's registrar database, and UA1 learns from the 200 OK message that UA2 resides in the IPv4 domain. The call setup procedure in Figure 3.6 is described as follows.

Step 1. UA1 sends an IPv6 SIP INVITE message to UA2 through the SIP server. Suppose that the port number filled in the SDP description is Z. UA1 prepares to use

its IPv6 transport address (i.e., its IPv6 address and the port number Z) for sending/receiving RTP packets. Also, UA1 prepares to use its IPv4 transport address (i.e., its IPv4 address and the port number Z) for RTP in case UA2 supports IPv4 only.

Step 2. When the SIP server receives the INVITE message, it performs IP version comparison. Since the versions of UA1's and UA2's IP addresses are different, the SIP server translates the INVITE message into an IPv4 SIP INVITE message that is the same as an IPv4 SIP INVITE message sent by UA1. Specifically, the SIP server retrieves UA1's IPv4 contact information from its registrar database. It then replaces UA1's IPv6 contact information in the *Via* and the *Contact* header fields by UA1's IPv4 contact information. In addition, UA1's IPv6 address in the SDP description is replaced by UA1's IPv4 address. However, the port number in the SDP description is unchanged. The translated IPv4 INVITE message is then sent to UA2.

Step 3. When UA2 receives the INVITE message, it accepts the call by sending an IPv4 200 OK message to UA1 through the SIP server.

Step 4. Upon receipt of the 200 OK message, UA1 learns that UA2 is in the IPv4 domain. Therefore, an IPv4 ACK message is sent to UA2 through the SIP server, and UA1 chooses its IPv4 transport address (i.e., its IPv4 address and the port number Z mentioned in Step 1) for sending/receiving RTP packets.

After the call is established, IPv4 RTP packets are delivered between UA1 and UA2.

3.5 Comparisons

In this section, the four solutions discussed in this chapter are compared in several aspects listed in Table 3.1 and described as follows.

UA Modification: In the SIP-ALG solution, standard UAs are used. In the Redirect solution, the calling party is equipped with the redirect function from IPv6 to IPv4.

Table 3.1: Comparisons of the IPv4/IPv6 Interworking Solutions for SIP-based VoIP

Solution	UA Modification	SIP Server Modification	Call Setup Complexity	RTP Transmission Latency
SIP-ALG Solution	Calling Party: No Called Party: No	Yes	6 Messages with 3 Translations	High
Redirect Solution	Calling Party: Yes Called Party: No	Yes	9 Messages	Low
ICE Solution	Calling Party: Yes Called Party: Yes	No	10 Messages	Low
Our Solution	Calling Party: Yes Called Party: No	Yes	6 Messages with 1 Translation	Low

The called party does not require any modification. In the ICE solution, both the calling and the called parties support ICE. Also, the called party should be able to process the SIP messages that contain IPv6 addresses. In our solution, the calling party needs to select the IP version for RTP based on the 200 OK message sent from the called party. No modification is made to the called party.

SIP Server Modification: In the SIP-ALG solution, the SIP server is equipped with IP version comparison function and SIP-ALG. In the Redirect solution, the SIP server needs to perform IP version comparison and to act as a redirect server. In the ICE solution, the SIP server encapsulates the SIP messages in IPv4 or IPv6 packets based on the destination of the SIP messages, which is determined according to the standard SIP message processing procedure defined in [23]. This task does not require any modification to the SIP server. In our solution, the SIP server performs IP version comparison and translates the IPv6 INVITE message into an IPv4 INVITE message based on the calling party's contact information.

Call Setup Complexity: Compared to the standard SIP call setup procedure, the SIP-ALG solution incurs additional call setup overhead for the SIP message translation (between IPv6 and IPv4) performed at the SIP server. The Redirect solution requires an extra INVITE transaction (including INVITE, 302 Moved Temporarily, and ACK messages). In the ICE solution, extra call setup overhead is introduced to carry out the connectivity checks. In our solution, the SIP server needs to translate

the first IPv6 INVITE message into an IPv4 INVITE message. The numbers of the call setup messages and the SIP message translations of all solutions are listed in Table 3.1.

RTP Transmission Latency: In the SIP-ALG solution, the RTP packets are delivered between the call parties indirectly through the RTP proxy. In the other solutions, the RTP packets are directly delivered between the call parties. We have conducted experiments to measure the extra RTP transmission delay introduced by the RTP proxy. In our experiments, the RTP proxy is a PC (with an Intel Pentium 4 3.0GHz CPU and 1.0GB main memory) installed with RTPProxy [1] on Linux operating system. The RTP proxy connects to an IP network through 100 Mbps Ethernet. During the experiments, RTP packets are injected to the RTP proxy at the rate of 10 Mbps. Each of the RTP packets contains 20 ms voice data encoded in G.711 (172 bytes in length). The average latency of translating an RTP packet is 5.9 ms. This extra delay is more than twice the network latency in most cases we have investigated. For example, the network latency from National Chiao Tung University to ArtDio VoIP operator (passing through 7 routers) is 2.012 ms in average. Therefore, the RTP transmission latency in the SIP-ALG solution is about 6 to 8 ms more than that in the other solutions.

3.6 Summary

This chapter described three existing IPv4/IPv6 interworking solutions for SIP-based VoIP: the SIP-ALG solution, the Redirect solution, and the ICE solution. Then we proposed an effective solution and compared the new approach with the three existing solutions. Our study indicated that the RTP transmission performance in the SIP-ALG solution is worse than that in the other three solutions. Both the Redirect solution and the ICE solution introduce extra call setup messages with extra call setup overhead. In our solution, the least extra call setup overhead is introduced, and no impairment is incurred to the RTP transmission performance.

Chapter 4

Conclusions

Since every UA requires a unique IP address, the insufficiency of IPv4 addresses becomes a problem in SIP-based VoIP deployment. This problem can be solved by either introducing NATs into IPv4 networks or replacing IPv4 with IPv6. In Chapter 2, we discussed the case where NATs are adopted. An NAT is used so that a public IPv4 address is shared by multiple UAs in the private network. Nevertheless, the NAT translates the IP-layer transport address but leaves the application-layer content unchanged, which results in inconsistency between the transport address at the network and the transport layers and that in the SIP layer. Five UA-based and two server-based widely used solutions were discussed. Also, we designed and developed a UA to analyze these solutions in terms of UA modification, scope of NATs supported, multi-layer NAT traversal, extra server/public IP address requirements, configuration complexity, and time complexities. At last, we concluded that there is no SIP/RTP NAT traversing solution that is better than the others in all aspects.

In Chapter 3, the usage of IPv6 was discussed. IPv6 provides large address space, which avoids the usage of the NAT. However, IPv4/IPv6 interworking problem is occurred when IPv4/IPv6 dual-stack UAs attempt to interact with IPv4-only UAs using IPv6. Three existing solutions were described. Also, we proposed an effective solution by performing SIP message translation to the INVITE message only. The four solutions were evaluated in terms of UA/SIP server modification, call setup complexity, and RTP transmission latency. It was shown that our solution outperforms the other solutions in the call setup and the RTP transmission performance. In summary, the usage of IPv6 coop-

erating with our proposed solution is the best choice with the most efficient performance for deploying SIP-based VoIP services.

Bibliography

[1] RTPProxy – RTP (Realtime Transport Protocol) Proxy Server. http://ftp.iptel.org/pub/rtpproxy/.

[2] Snom VoIP-telephones, Solutions for NAT Traversal in SIP Environment. http://www.iptel.org/info/products/etc/snom-stun.pdf.

[3] The eXtended osip Library. http://savannah.nongnu.org/projects/exosip/.

[4] UPnP™ Forum. http://www.upnp.org/.

[5] UPnP™ Internet Gateway Working Committee. http://www.upnp.org/newsletters/newsletterI/committee.htm#Internet.

[6] The Point-to-Point Protocol (PPP). RFC 1661, IETF, July 1994.

[7] Microsoft, What Is VPN - Technologies Related to VPN. http://www.microsoft.com/technet/prodtechnol/windowsserver2003/library/TechRef/bc5c01ea-189e-4d86-a9b7-d426ac80c8a2.mspx, March 2003.

[8] M. Borella, D. Grabelsky, J. Lo, and K. Taniguchi. Realm specific IP: protocol specification. RFC 3103, IETF, October 2001.

[9] G. Camarillo, R. Penfield, A. Hawrylyshenm, and M. Bhatia. Requirements from SIP (Session Initiation Protocol) Session Border Control Deployments. Internet-Draft, October 2006. draft-camarillo-sipping-sbc-funcs-05.

[10] E. M. Castro, T. Miguel, and S. Pavon. Transition of Applications to IPv6. *Upgrade Journal*, 6(2):15–18, April 2005.

[11] W.-E. Chen, Y.-B. Lin, and A.-C. Pang. An IPv4-IPv6 Translation Mechanism for SIP Overlay Network in UMTS All-IP Environment. *IEEE Journal of Selected Areas in Communications*, 23(11):2152–2160, November 2005.

[12] W.-E. Chen, Y.-H. Sung, and Y.-B. Lin. SIPv6 Analyzer: An Analysis Tool for 3GPP IMS Services. *Wireless Communications and Mobile Computing Journal.*

[13] S. Deering and R. Hinden. Internet protocol, version 6 (IPv6) specification. RFC 2460, IETF, December 1998.

[14] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, IETF, June 1999.

[15] Y. Y. Goland, T. Cai, P. Leach, Y. Gu, and S. Albright. Simple Service Discovery Protocol/1.0. Internet-Draft, April 1999. draft-cai-ssdp-v1-03.txt.

[16] M. Handley and V. Jacobson. SDP: Session Description Protocol. RFC 2327, IETF, April 1998.

[17] J. Mulahusic and H. Persson. SIP Issues in Dual-stack Environments. Internet-Draft, February 2003. draft-persson-sipping-sip-issues-dual-stack-00.

[18] S. Olson, G. Camarillo, and A. Roach. Support for IPv6 in Session Description Protocol (SDP). RFC 3266, IETF, June 2002.

[19] J. Rosenberg. Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. Internet-Draft, March 2007. draft-ietf-mmusic-ice-15.

[20] J. Rosenberg, C. Huitema, R. Mahy, and D. Wing. Simple Traversal of UDP Through Network Address Translators (NAT) (STUN). Internet-Draft, October 2006. draft-ietf-behave-rfc3489bis-05.

[21] J. Rosenberg, R. Mahy, and C. Huitema. Obtaining Relay Addresses from Simple Traversal of UDP Through NAT (STUN). Internet-Draft, October 2006. draft-ietf-behave-turn-02.

[22] J. Rosenberg and H. Schulzrinne. An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing. RFC 3581, IETF, August 2003.

[23] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, IETF, June 2002.

[24] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, IETF, July 2003.

[25] M-K. Shin, Y-G. Hong, J. Hagino, P. Savola, and E. M. Castro. Application Aspects of IPv6 Transition. RFC 4038, IETF, March 2005.

[26] D. Sisalem, J. Fiedler, and R. Ruppelt. SIP and IPv6: Why and How. *SAINT2003, Orlando (Florida, USA)*, pages 27–31, January 2003.

[27] P. Srisuresh and M. Holdrege. IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663, IETF, August 1999.

[28] P. Srisuresh, J. Kuthan, J. Rosenberg, A. Molitor, and A. Rayhan. Middlebox communication architecture and framework. RFC 3303, IETF, August 2002.