國 立 交 通 大 學

資 訊 工 程 學 系

碩 士 論 文

為 SIP 應用程式解決穿越網路位址轉換機的問題

A NAT traversal solution for SIP applications

指導教授：張明峰 教授

研 究 生：陳建彰

中 華 民 國 九 十 四 年 六 月

為 SIP 應用程式解決穿越網路位址轉換機的問題

A NAT traversal solution for SIP applications

研　究　生： 陳建彰　　　　　Student: Chien-Chang Chen

指導教授： 張明峰教授　　　　Advisors: Prof. Ming-Feng Chang

國 立 交 通 大 學
資 訊 工 程 學 系
碩 士 論 文

A Thesis Submitted to
Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of Master
in
Computer Science and Information Engineering
June 2005
Hsinchu, Taiwan, Republic of China

中華民國九十四年六月

為 SIP 應用程式解決穿越網路位址轉換機的問題

學生：陳建彰　　　　　　　　　　　　　指導教授：張明峰 博士

國立交通大學資訊工程學系（研究所）碩士班

中文摘要

網路位址轉換機(Network Address Translator)普遍地用來解決 IPv4 位址不夠用的瓶頸；但它卻會阻礙某些應用程式或是多媒體的傳輸，這叫做「穿越網路位址轉換機的問題」。許多人研究出解決這個問題的方法，尤其是針對對稱式的網路位址轉換機，因為它是當中最難克服的一種。互相建立連線機制(Interactive Connectivity Establishment)提供一個不錯的方法可以解決所有種類的穿越問題。但我們發現到他的作法缺乏效率。

於本論文中，我們提出一種想法，就是在註冊的過程中完成大部份穿越網路位址轉換機的前置工作，來減少建立通話時檢驗路由位址可否連線的負擔，比互相建立連線機制花費更少的工作去達成相同的功能。我們也主張這個架構應該以一個類似點對點(Peer-to-Peer)的模型為基礎藉此消去對轉送伺服器的需求，以及降低對註冊伺服器的負擔，諸如此類優點。「穿洞」在我們的架構上扮演一個很重要的角色，它的意思是讓網路位址轉換機內部的主機送封包給外部的主機藉此在網路位址轉換機上製造出相對應的接口。我們也認為應該在註冊的過程中打出要給訊號端用的洞；但是在建立通話之前再打出多媒體資料要用的洞。這樣可以減去和網路位址轉換機上的接口保持聯繫、延續它的有效時間的負擔。

A NAT traversal solution for SIP applications

Student: Chien-Chang Chen Advisor: Dr. Ming-Feng Chang

Department of Computer Science and Information Engineering

National Chiao Tung University

Abstract

Network Address Translator (NAT), which is used to extend the IPv4 address space, may break Internet connections and media transmission. This is refered to as “NAT traversal problem”. Many researches have investigated this problem, especially for the symmetric NAT, which is the most difficult to traverse. Interactive Connectivity Establishment (ICE) provides a pretty good choice to solve the traversal problem, but there are some redundant steps that can be simplified for VoIP communications.

In the thesis, we present a solution that accomplishes most tasks to traverse NAT during VoIP user registration, and reduces the cost in verifying allocated addresses. Our method efficiently solves each type of NAT traversal problem with less overhead than the ICE. Our NAT traversal solution is also based on a P2P-like model to eliminate the need of dedicated relay servers and to reduce the load of the SIP registrar. Hole punching, which means that an internal host sends packets to an external host to create a mapping port on the NAT, plays a significant role in our solution; a SIP UA punches a hole on the NAT for exchanging signaling during registration, and punches another for media transmission just before establishing a call. Thus our method reduces the overhead of keep-alive messages.

誌謝

感謝我的指導老師，張明峰教授，兩年來細心及費心的指導。方能順利完成此篇論文。於受業間，老師的耐心指正和灌輸我正確的人生方向，亦是讓我受益良多。在撰寫論文期間，感謝弘鑫學長在整體架構上的意見提供。感謝榮泰和其範和瑋晟這三位同儕的彼此砥礪和互相扶持，一起走過這研究所的兩年生活。也要感謝 VoIP 實驗室的學弟們，雅智、詩賢、至鴻和俊賢的支持和鼓勵。最後要感謝父母的全力栽培，讓我無後顧之慮的可以全力在學業上衝刺。養育之恩，謹記在心。

最後，原將此篇論文獻給我所愛的家人與好友。

Tables of Contents

List of Figures

1896

List of Tables

Chapter 1 Introduction

In recent years, Voice-over-IP (VoIP) has become popular and many VoIP products have been developed. VoIP phones enable people call each other through the Internet in the same way as they dial a PSTN phone call, and reduce the cost for telephony. This is the main advantage of VoIP. However, firewall and Network Address Translator (NAT) [1] traversal problem makes VoIP applications difficult to use in private intranets. Numerous researches have been conducted to overcome the problem. In this thesis, we present a new mechanism that solves NAT traversal without modifying NAT devices, and considers all cases of call setup and their optimal solutions. We integrate Session Initiation Protocol (SIP) [2] registration with relay server assignment, SIP-based NAT type detection, hole punching, and RTP port allocation and so on. Hole punching is the procedure that an internal host of a NAT sends a packet to external host, and produces a mapping port on the NAT.

1-1. Related work

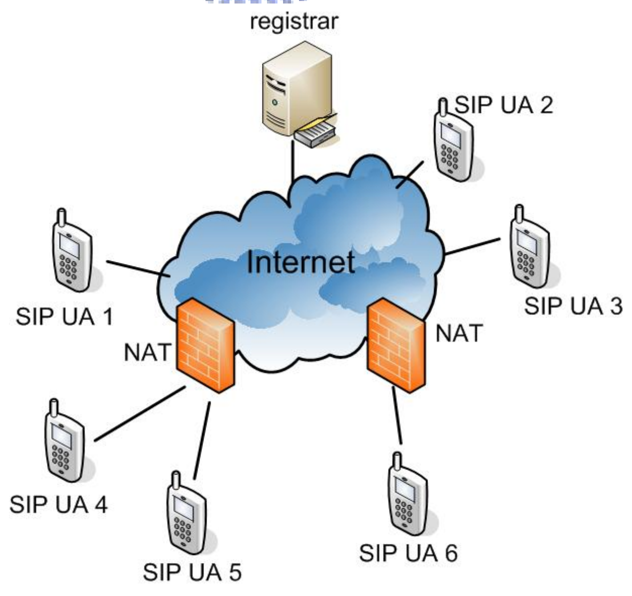


Figure 1-1: SIP architecture

SIP is a signaling protocol handling setup, modification, and tear-down of multimedia sessions. SIP was defined in Request for Comments (RFC) 2543 by a working group of Internet Engineering Task Force (IETF), and a new SIP RFC has been produced, RFC number 3261. Figure 1-1 describes the SIP architecture with NAT devices. SIP is a client-server protocol. A registrar accepts registration requests from SIP UAs and maintains their information. A SIP UA is a client that can be used to set up multimedia communications over the Internet, and can be installed on a PC, a PDA, or a SIP phone.

1-2. NAT traversal problem and possible solutions

■ NAT traversal problem

NAT was originally developed as a short-term solution to combat IPv4 address exhaustion. It allows globally registered IP addresses to be reused or shared by several hosts. The classic NAT defined by RFC 1631 [3] maps IP addresses from one realm to another. Although it can be used to translate between any two address realms, NAT is most often used to map IP address from the non-routable private address defined by RFC 1918 [4].

NAT, while providing many benefits, also comes with many drawbacks. The most troublesome of those is the fact that they break many existing IP applications and make it difficult to deploy new ones.

In RFC 3489 [5], NAT were grouped into four types: full-cone NAT, restricted cone NAT, port-restricted cone NAT, and symmetric NAT.

(1) Full cone: a full cone NAT is one where all requests from the same internal IP address and port are mapped to the same external IP address and port. Furthermore, any external host can send a packet to the internal host, by sending a packet to the mapped external address.

(2) Restricted cone: a restricted cone NAT is one where all requests from the same internal IP address and port are mapped to the same external IP address and port. Unlike a full cone NAT, an external host (with IP address X) can send a packet to the internal host only if the internal host had previously sent a packet to IP address X.

(3) Port Restricted cone: a port restricted cone NAT is like a restricted cone NAT, but the restriction includes port numbers. Specifically, an external host can send a packet, with source IP address X and source port P, to the internal host only if the internal host had previously sent a packet to IP address X and port P.

(4) Symmetric: a symmetric NAT is one where all requests from the same internal IP address and port, to a specific destination IP address and port, are mapped to the same external IP address and port. If the same host sends a packet with the same source address and port, but to a different destination, a different mapping is used. Furthermore, only the external host that receives a packet can send a UDP packet back to the internal host.

■ Possible solutions

In this section, we don't consider solutions, such as ALG [6], UPnP, Middlebox [7], that solve the traversal problem by modifying the NAT itself. The number of existing NAT is too huge.

SIP extension for NAT traversal: it allows for SIP responses to go back to the source port of the request and allows a registrar to get the source IP/port instead of the "Contact" in a REGISTER. We consider that this extension just helps the development of traversal solution, rather than a completed solution.

Simple Traversal of UDP through NAT (STUN): the protocol allows entities behind a NAT to discover the presence of a NAT and its type, and then to learn the addresses bindings allocated by the NAT. STUN requires no changes to NAT devices, and works with an arbitrary number of NATs in tandem between the application entity and the public

Internet. However, STUN can't resolve the case of symmetric cone NAT.

Traversal Using Relay NAT (TURN) [8]: TURN allows a client to relay packets through a server that resides on the public Internet. Although TURN will almost always provide connectivity to a client, it comes at high cost to the provider of TURN server. It is therefore desirable to use TURN as a last resort only, preferring other mechanisms (such as STUN or direct connectivity) when possible.

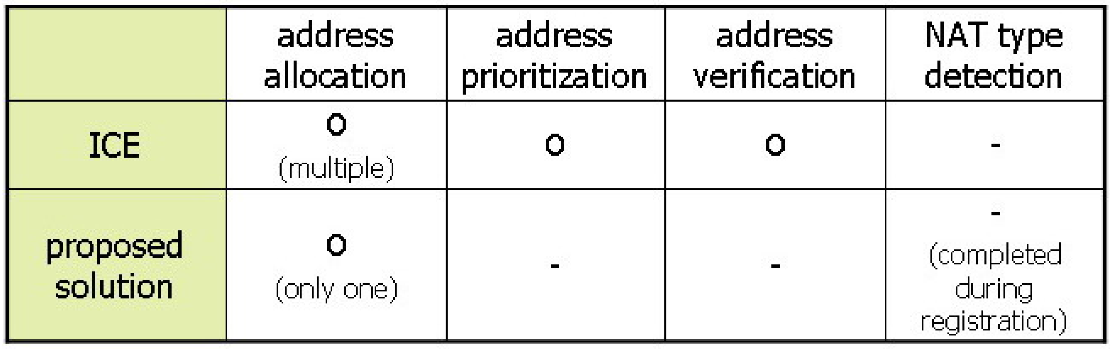
Interactive Connectivity Establishment (ICE) [9]: ICE is not a new protocol. It makes use of both STUN and TURN protocols, but uses them in a specific methodology which avoids many of the pitfalls of using any one alone. The key idea behind ICE is that network conditions and connectivity assumptions can, and will change. Therefore ICE clients have no need to know whether they are behind a NAT, its type, or network topology. ICE always will find a means for communicating if one physically exists.

An ICE client has many addresses where it can receive media, such as local IP address, addresses learned from a STUN server, or a TURN server. It has to gather as many addresses as possible and includes them in its INVITE message, and decides the priority of addresses based on lowest cost and maximum Qos before making a call. When the called party receives the INVITE message, it will do the same allocation steps as caller, and verify the connectivity of each address in the receiving message and choose the appropriate address to send media sessions based on the priority.

The advantage of ICE is that clients have no need to know the NAT type and network topology. It solves all kinds of NAT traversal problem with the cheapest path for service provider. However, we find some unnecessary overhead in ICE. First, we think that there are no needs to do so many tasks when establishing calls. If we assure the NAT type and route address ahead of time, for example: during registration, we don't need allocate more than one addresses and verify their connectivity when making a call. Second, if we choose to implement ICE protocols, we must also need to maintain STUN servers and TURN servers in the public network for clients. Third, ICE clients needs to serve as a STUN server when the called party wants to verify the connectivity of addresses.

1-3. The solution of the thesis

The following table presents the comparison of call setup overhead between ICE and the proposed solution in the thesis:



	address allocation	address prioritization	address verification	NAT type detection
ICE	o (multiple)	o	o	-
proposed solution	o (only one)	-	-	- (completed during registration)

Table 1-1: call setup overhead: ICE vs. proposed solution

In the thesis, a SIP UA will detect the presence of NAT and its type, and assures route address during registration. Thus the UA does not need to check the connectivity of route address. In other words, we can eliminate the procedures of address prioritization and verification when making calls and reduce overhead further.

1-4. Overview of the thesis

The rest of this thesis is organized as follows. Chapter 2 describes the overall NAT traversal mechanism. Chapter 3 describes how to design and implement such a system presented in the previous chapter. At last, Chapter 4 gives conclusions and discusses future work for the thesis.

Chapter 2 Mechanism of NAT Traversal

2-1. Overview of the proposed mechanism

The chief purpose of the proposed mechanism is providing efficiency and optimization for SIP UAs behind a NAT to make calls with others. The principle is “do not use relay server if possible” and “most tasks in registration, fewer tasks in call setup”. We make necessary provision during registration, and this makes call establishment easier. We extend the concept with a Peer-to-Peer(P2P)-like model (we call it P2P-like because there are no P2P algorithms, such as Chord [10], Content-Addressable Network (CAN) [11] in our mechanism, but there are no centralized servers except for a SIP registrar either.) which makes each SIP UA provide the functions of relay server further. The advantage of the P2P-like model involves that service providers don’t need to maintain dedicated relay servers, and it reduces the overhead of registrar, and provides robustness. We also consider that the SIP UA behind a NAT, whatever its type, must establish calls with lowest cost. That is, sometimes we may need a relay server to transfer only signaling data in port-restricted cone case, sometimes both signaling and media data in symmetric case or even we need no relay server while in full cone case.

First, we define an entity: SN, a super node, is a SIP UA assigned by registrar to serve the UA behind a NAT. It still can make a call while serving others. The basic requirement for a SN is that it must be in the public network, because it has to relay signaling or media data for the UA behind a NAT.

The functions of SN include type detection of NAT, responding keep-alive message, allocating real-time transport protocol (RTP) [12] ports, maintaining UA table, routing signaling and media sessions. The functions of the UA behind a NAT involve SIP registration, type detection of NAT, maintaining UA table and sending keep-alive request to its SN. The functions of registrar involve registration, assigning SN for UAs behind a NAT, detection of NAT, and maintaining database.

2-2. Registration

SIP UAs must register before making calls with anyone. Because much key information can be collected and exchanged during registration.

2-2-1. Case 1: SIP UA in the public network

In the first case, we present a registration procedure for the UA in the public network. There is NAT detection in this registration procedure and the SIP-URI must be filled with IP address, instead of host name.

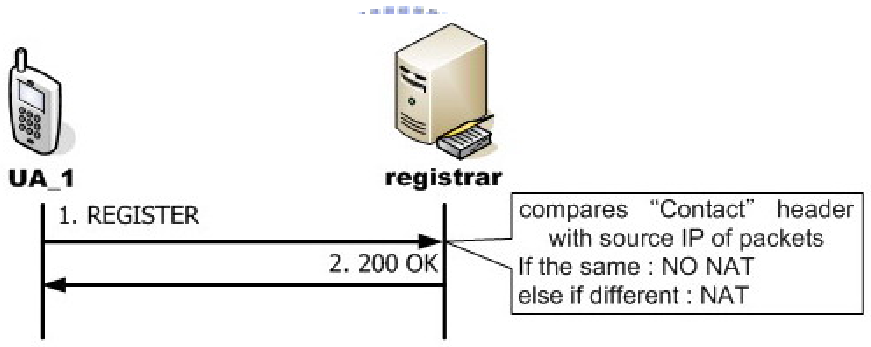


Figure 2-1: registration case 1 - SIP UA in the public network

1. REGISTER：UA_1, a SIP UA, sends a SIP REGISTER message to its registrar.

2. 200 OK：When the registrar receives the REGISTER message, it would check the source of packets and the "Contact" header in the REGISTER message. If UA_1 is in the public network, these two addresses will be the same (this is because the source of packets will be the IP address of NAT if the source host is behind a NAT, and the "Contact" header will be a private IP address).

Thus registrar can identify UA_1 is in the public network and returns 200 OK response to UA_1. So far, Registration is complete.

2-2-2. Case 2: SIP UA behind a full cone NAT

In the second case, we show the registration procedure for the UA behind a full cone NAT.

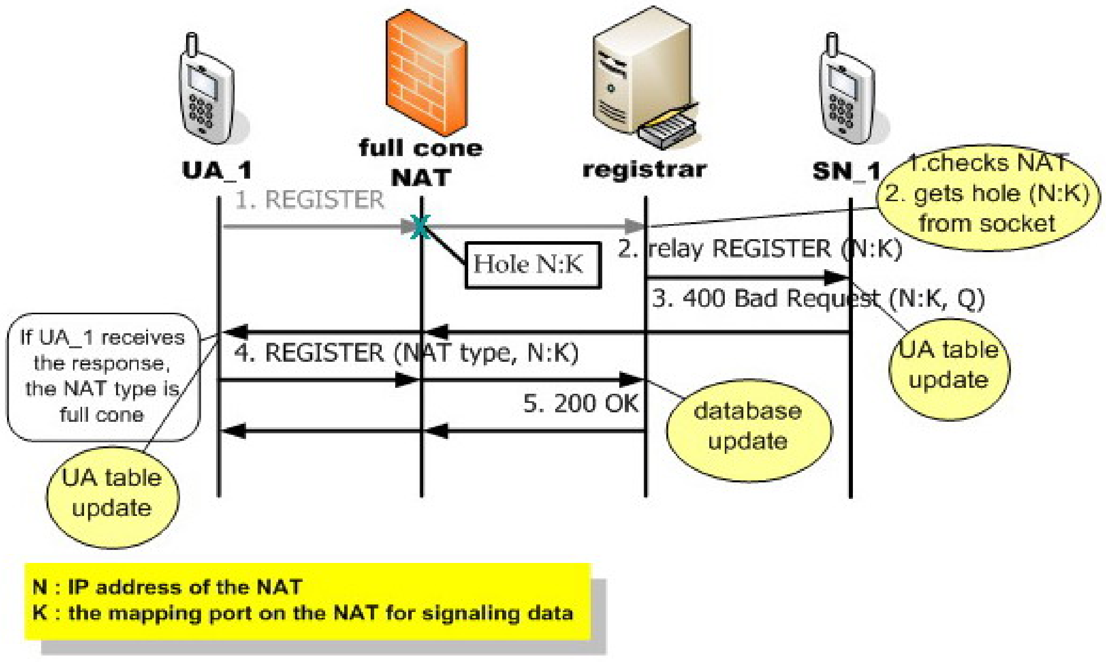

UA_1
full cone NAT
registrar
SN_1
1. REGISTER
Hole N:K
2. relay REGISTER (N:K)
1.checks NAT 2. gets hole (N:K) from socket
3. 400 Bad Request (N:K, Q)
UA table update
4. REGISTER (NAT type, N:K)
5. 200 OK
database update
If UA_1 receives the response, the NAT type is full cone
UA table update
N : IP address of the NAT
K : the mapping port on the NAT for signaling data

Figure 2-2: registration case 2 - SIP UA behind a full cone NAT

1. REGISTER： UA_1, a SIP UA, sends a SIP REGISTER message to its registrar, and registrar will checks the source of the socket with the “Contact” header in the REGISTER message. If UA_1 is behind a NAT, these two addresses would be different.

On the other hand, there is a “hole” punched by the REGISTER message. We name the hole “N:K”, which N means the IP address of the NAT, and K means the mapping port on the NAT.

2. REGISTER： Registrar finds that UA_1 is behind a NAT and chooses a SN from the database to serve UA_1 based on some policies. Then it relays REGISTER message to SN_1, which is a SN. Besides, registrar would modify the relayed message. The message has to carry the hole, “N:K”.

3. 400 Bad Request：SN_1 receives the REGISTER message. Then it fetches the “N:K” from the message, and sends a “400 Bad Request” response which the destination is the “N:K”. This is the “first” detection of NAT type to identify the NAT is a full cone or not. Based on the characteristic of full cone NAT: “any external host can send a packet to the internal host from the same mapping port on the NAT”, if UA_1 receives the message, the NAT will be a full cone.

The response has to carry the IP address and port of SN_1. Furthermore, SN_1 would allocate a port for RTP transfer future and the response must carry it, too.

4. REGISTER：UA_1 receives the response from SN_1. Then it will send the second REGISTER message to registrar to return information, including “NAT type” and “N:K”. The purpose of “N:K” is to route packets into NAT when someone wants to establish a call with UA_1 and sends INVITE to registrar future. For a full cone NAT, a hole can be reused. .

5. 200 OK：Registrar receives REGISTER message and updates the database. Registration is complete.

2-2-3. Case 3: SIP UA behind a non full cone (restricted cone, port-restricted cone, or symmetric) NAT

In the third case, we show a registration procedure for the UA behind a non full cone NAT. In other words, it may be restricted cone, or port-restricted cone, or symmetric.

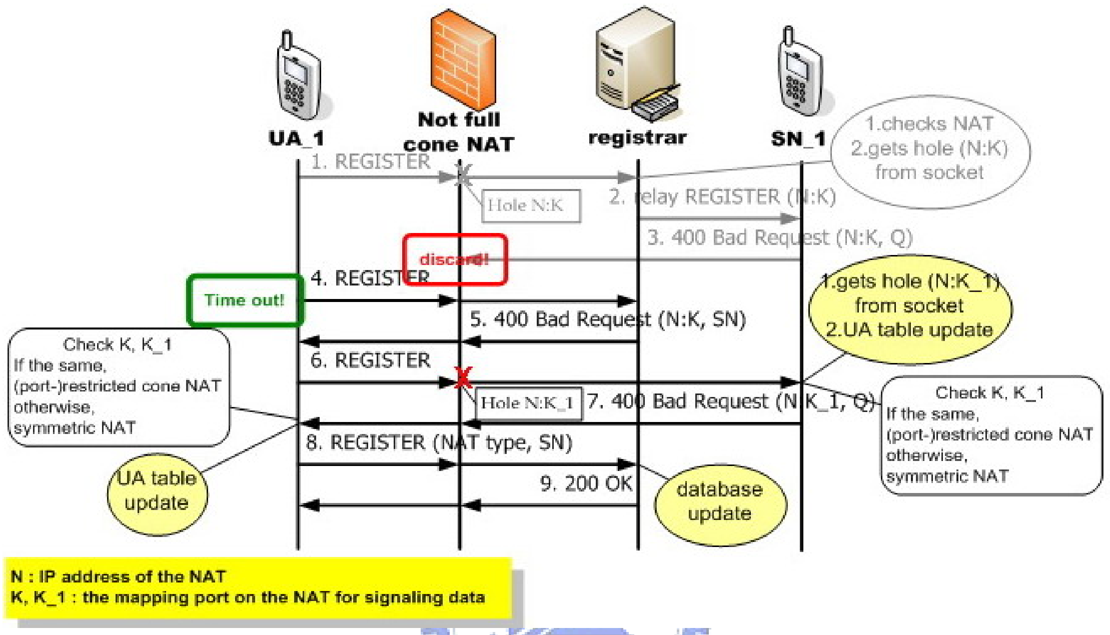

UA_1
Not full cone NAT
registrar
SN_1
1.checks NAT
2.gets hole (N:K) from socket
1. REGISTER
Hole N:K
2. relay REGISTER (N:K)
3. 400 Bad Request (N:K, Q)
discard!
Time out!
4. REGISTER
1.gets hole (N:K_1) from socket
2.UA table update
5. 400 Bad Request (N:K, SN)
Check K, K_1
If the same, (port-)restricted cone NAT
otherwise, symmetric NAT
6. REGISTER
Hole N:K_1
7. 400 Bad Request (N:K_1, Q)
Check K, K_1
If the same, (port-)restricted cone NAT
otherwise, symmetric NAT
8. REGISTER (NAT type, SN)
UA table update
9. 200 OK
database update
N : IP address of the NAT
K, K_1 : the mapping port on the NAT for signaling data

Figure 2-3: registration case 3 - SIP UA behind a non full cone (restricted cone, port-restricted cone, or symmetric) NAT

1. REGISTER：The same as “**1. REGISTER**” in Case 2.

2. REGISTER：The same as “**2. REGISTER**” in Case 2.

3. 400 Bad Request：The same as “**3. 400 Bad Request**” in Case 2.

4. REGISTER：If UA_1 is behind a NAT but the type is not full cone, it won’t receive the “400 Bad Request” response from SN_1, and the retransmitting timer in UA_1 will fire. Then UA_1 sends SIP REGISTER to registrar again.

5. 400 Bad Request：When registrar receives the second REGISTER from UA_1, it will realize that UA_1 is behind a non full cone NAT. Registrar will send “400 Bad

Request” response to UA_1 and assign a SN, SN_1, for UA_1. In addition to the IP address and port of SN, the response will carry “N:K”, which is the hole between UA_1 and registrar. The purpose of “N:K” is to detect NAT type later.

6. **REGISTER**：UA_1 receives “400 Bad Request” response from registrar and get the IP address of SN_1 and “N:K”. Then it will send a REGISTER message to SN_1 to punch a new hole.

“N:K” is useless for routing data into NAT because the characteristic of restricted cone or symmetric NAT: “an external host can send a packet to the internal host only if the internal host had previously sent a packet to the external host”, “N:K” can not be reused by other UAs if the NAT is not full cone.

7. **400 Bad Request**：SN_1 receives the REGISTER message and gets the “hole” from socket, we name the new hole “N:K_1”. So far, SN collects two “holes” in the NAT: one is “N:K” and the other is”N:K_1”. We can identify the NAT type by comparing these two “holes”: if they are the same, the NAT type will be “restricted cone” or “port-restricted cone”; on the contrary, if they are different, the NAT type will be exactly “symmetric”. This is because the characteristic of symmetric NAT:”if the same host sends a packet with the same source address and port, but to a different destination, a different mapping is used.” That’s the reason that “N:K” and “N:K_1” will be different if the NAT is symmetric.

In this step, SN must allocate a RTP port for RTP relay later. We name the port “Q”. SN_1 will return “400 Bad Request” response to UA_1 carrying “N:K_1” and “Q”.

8. **REGISTER**：UA_1 receives “400 Bad Request” response from SN_1, and records the “N:K_1” and “Q”(UA table update). UA_1 will identify the NAT type in the same way. After that, UA_1 will send REGISTER message to registrar again to return

information, involving “NAT type”, and “N:K_1” if the NAT is a restricted cone or port-restricted cone, or the “IP address and port of SN” if the NAT is a symmetric.

9. 200 OK：Registrar receives REGISTER message and updates its database. Registration is complete.

■ Keep alive

After registration, UAs behind a NAT must keep alive (means sending some packets outward the NAT to someone in the public network and receiving the response periodically) with its SN to refresh the binding time of “hole” on the NAT. So the period of keep-alive message must shorter than lifetime. For a full cone NAT, the binding lifetime is 5 minutes; while for a restricted or symmetric NAT, the binding lifetime is 30 seconds.

2-3. Call setup

■ <u>Hole punching for media sessions</u>

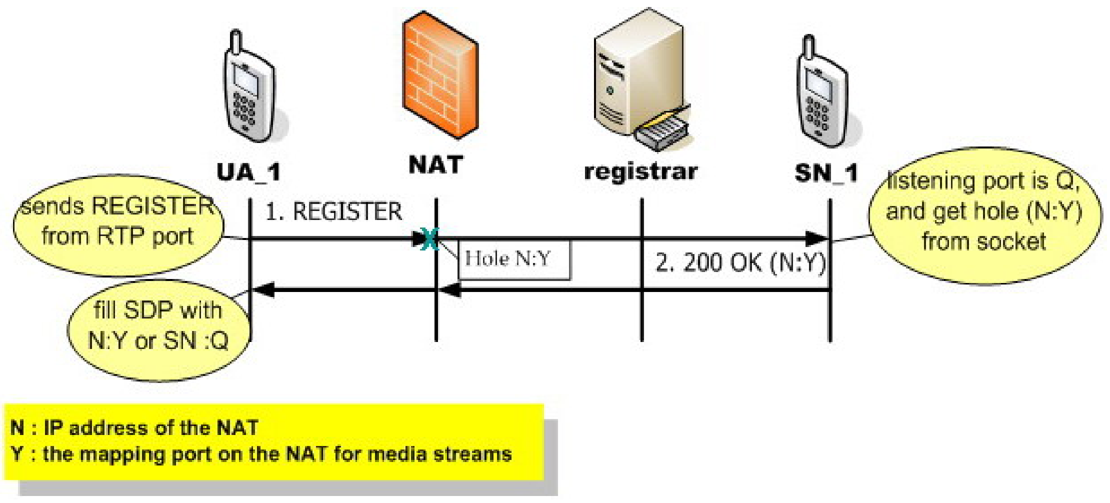

UA_1
NAT
registrar
SN_1
sends REGISTER from RTP port
1. REGISTER
Hole N:Y
2. 200 OK (N:Y)
listening port is Q, and get hole (N:Y) from socket
fill SDP with N:Y or SN :Q
N : IP address of the NAT
Y : the mapping port on the NAT for media streams

Figure 2-4：hole punching for media sessions

Before establishing a call, an UA behind a NAT must punch another hole for media sessions. The reason is that the signaling port and the media port in the UA are often different. For example, in our experiment the signaling port is 5060, but the media port is 9000. If an UA behind a NAT is the caller of a call, it should complete “hole punching for media sessions” before sending an INVITE message. If an UA behind a NAT is the callee of the call, it should complete “hole punching for media sessions” after receiving an INVITE message and before 200 OK being sent.

There is one advantage of starting the “hole punching for media sessions” just before making a call: we don’t need to keep alive for the instant “hole”. Thus we can reduce the overhead for UA and SN.

Like the process of hole punching for signaling data during registration, UA_1 sends a REGISTER message from its media port to the allocated RTP port: ”Q” on SN_1 preserved during registration. Then SN_1 receives the message and gets the “hole” from the socket.

We name it “N:Y”. SN_1 sends 200 OK response carrying “N:Y”. UA_1 receives 200 OK and fetches “N:Y”. It would write “N:Y” into the SDP attributes in the message later.

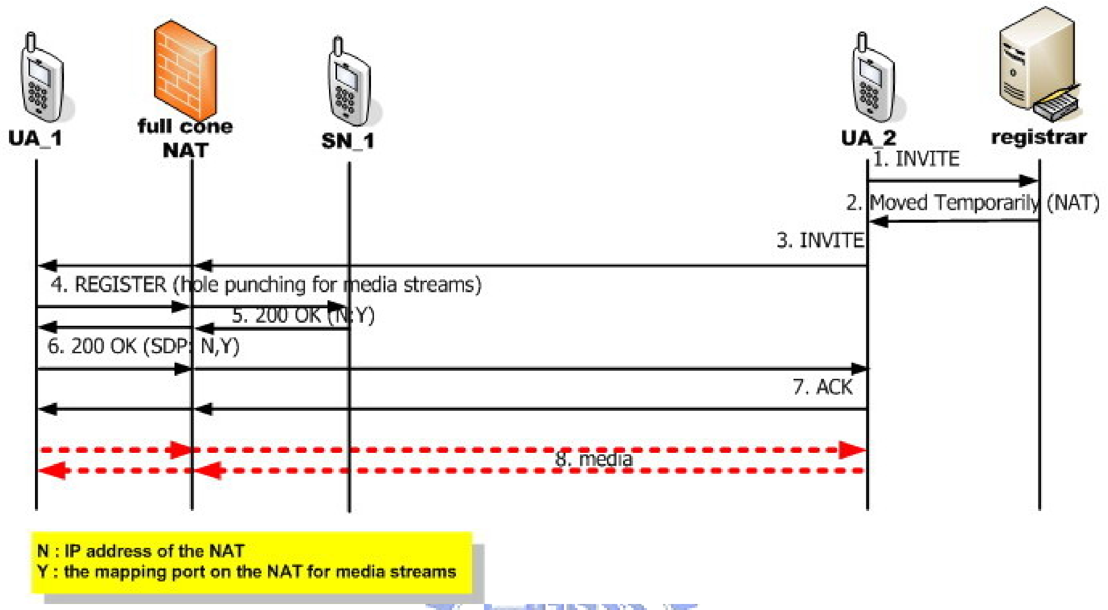
■ Optimal call setup solutions for each type of NAT

We focus on how to optimize the procedure of establishing a call between an UA behind a NAT and the other whatever it is in the public network or not. The design principle is: “less overhead and fewer messages when establishing a call”. Registrar, which is the only server in the mechanism, handles slight tasks based on the spirit of P2P. It’s just a redirect server when call setup occurs. Besides, there are no address allocation, prioritization, verification, or NAT detection for UAs to traverse NAT when establishing calls.

Generally speaking, solution 1 is for the UA behind a full cone NAT; solution 2 is for the UA behind a restricted or a port-restricted cone NAT; solution 3 is for the UA behind a symmetric NAT; solution 3’ is a special case, just for the condition that one is the UA behind a restricted or a port-restricted cone NAT and the other side is a symmetric NAT.

2-3-1. Solution 1: No signaling & media passes through the super node

The following figure is an example of solution 1: UA_2 wants to call UA_1 that is behind a full cone NAT.



UA_1
full cone NAT
SN_1
UA_2
registrar
1. INVITE
2. Moved Temporarily (NAT)
3. INVITE
4. REGISTER (hole punching for media streams)
5. 200 OK (N,Y)
6. 200 OK (SDP: N,Y)
7. ACK
8. media
N : IP address of the NAT
Y : the mapping port on the NAT for media streams

Figure 2-5: solution 1 example

1. **INVITE**：UA_2 sends INVITE message to its registrar. Each user who wants to setup a call with others will send INVITE to registrar first.

2. **302 Moved Temporarily**：Registrar receives the INVITE message, looks up its database for the location of the callee, and redirects the INVITE message to the NAT in front of UA_1.

3. **INVITE**：UA_2 receives the response and knows the location of the callee. It sends INVITE to the NAT.

4. **REGISTER (hole punching for media sessions)**： UA_1 receives INVITE message from UA_2. Then it will proceed the hole punching for media sessions. It sends REGISTER to its SN, SN_1.

5. **200 OK (hole punching for media sessions)**：SN_1 receives REGISTER message and fetches the "hole" from socket. We name it "N:Y". SN_1 sends 200 OK response carrying "N:Y" to UA_1.

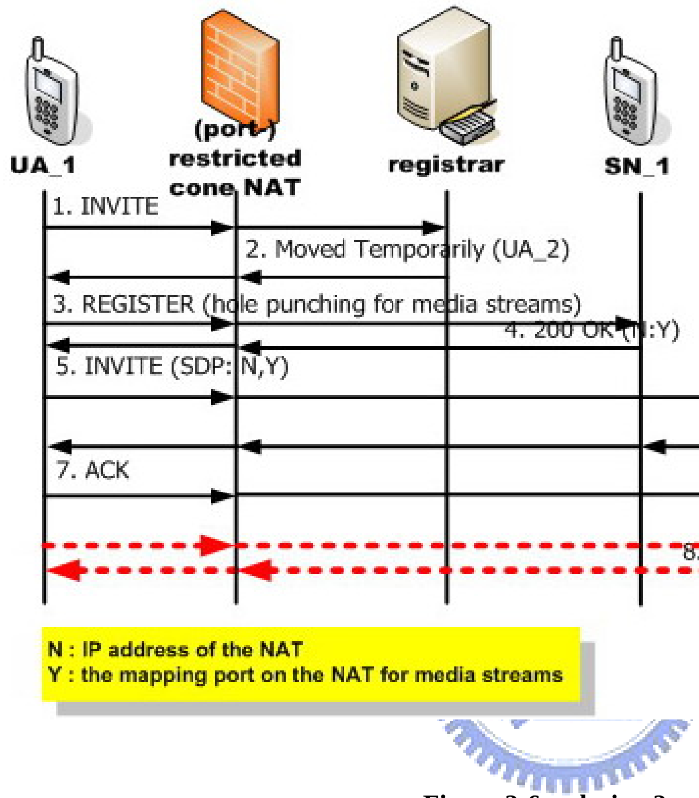
6. **200 OK**：UA_1 receives 200 OK. Then it returns 200 OK to UA_2. The Session Description Protocols (SDP) [13] attributes, "c" and "m", will be filled with N (IP address of NAT) and Y (mapping port on the NAT for media sessions).

7. **ACK**：UA_2 receives 200 OK and returns ACK back.

8. Call setup procedure finishes and both UA_1 and UA_2 begin to transfer media data. UA_2 will send media data to the NAT in front of UA_1 straightly.

2-3-2. Solution 2: Only inward signaling passes through the super node

The following figure is an example of solution 2: UA_1 behind a restricted cone NAT wants to call UA_2 in the public network.



UA_1
(port-) restricted cone NAT
registrar
SN_1
UA_2
1. INVITE
2. Moved Temporarily (UA_2)
3. REGISTER (hole punching for media streams)
4. 200 OK (N:Y)
5. INVITE (SDP: N,Y)
6. 200 OK
7. ACK
8. media
N : IP address of the NAT
Y : the mapping port on the NAT for media streams

Figure 2-6: solution 2 example

1. **INVITE**：UA_1 sends INVITE message to its registrar. Each user who wants to setup a call with others will send INVITE to registrar first.

2. **302 Moved Temporarily**：Registrar receives the INVITE message, looks up its database for the location of the callee, and redirects the INVITE message to the callee, UA_2.

3. **REGISTER (hole punching for media sessions)**：UA_1 receives the response and it will proceed the “hole punching for media sessions” first. It sends REGISTER to its SN, SN_1.

4. **200 OK (hole punching for media sessions)**：SN receives REGISTER message and fetches the “hole” from socket. We name it “N:Y”. SN_1 sends 200 OK response carrying

“N:Y” to UA_1.

5. **INVITE**：UA_1 gets the location of the callee. Because UA_2 is in the public network, UA_1 sends INVITE to UA_2 directly. The SDP attributes, “c” and “m”, will be filled with N (IP address of NAT) and Y (mapping port on the NAT for media sessions)

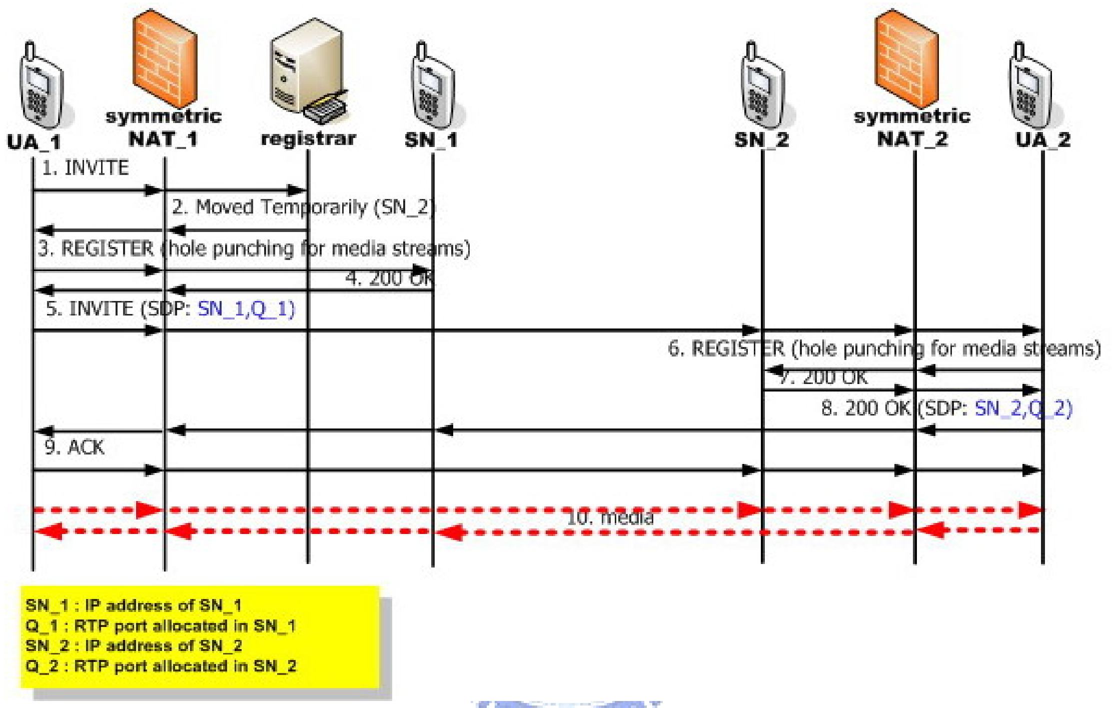
6. **200 OK**：UA_2 returns 200 OK to SN_1 by the “Via” header of previous INVITE message.

7. **ACK**：UA_1 receives 200 OK and returns ACK back.

8. Call setup procedure finishes and both UA_1 and UA_2 begin to transfer media data. UA_2 will send media data to the NAT in front of UA_1.

2-3-3. Solution 3: Inward signaling/inward media passes through the super node

The following figure is an example of solution 3: Both UA_1 and UA_2 are behind a symmetric NAT. UA_1 makes a call with UA_2.



UA_1
symmetric NAT_1
registrar
SN_1
SN_2
symmetric NAT_2
UA_2
1. INVITE
2. Moved Temporarily (SN_2)
3. REGISTER (hole punching for media streams)
4. 200 OK
5. INVITE (SDP: SN_1,Q_1)
6. REGISTER (hole punching for media streams)
7. 200 OK
8. 200 OK (SDP: SN_2,Q_2)
9. ACK
10. media
SN_1 : IP address of SN_1
Q_1 : RTP port allocated in SN_1
SN_2 : IP address of SN_2
Q_2 : RTP port allocated in SN_2

Figure 2-7: solution 3 example

1. **INVITE**：UA_1 sends INVITE message to its registrar. Each user who wants to setup a call with others will send INVITE to registrar first.

2. **302 Moved Temporarily**：Registrar receives the INVITE message, looks up its database for the location of the callee, and redirects the INVITE message to the SN of the callee, SN_2.

3. **REGISTER (hole punching for media sessions)**：UA_1 receives the response and it will proceed the "hole punching for media sessions" first. It sends REGISTER to its SN, SN_1.

4. **200 OK (hole punching for media sessions)**：SN_1 receives REGISTER message and fetches the “hole” from socket. SN_1 records the “hole” and sends 200 OK response to UA_1.

5. **INVITE**：UA_1 gets the location of the callee and sends INVITE to SN_2 because UA_2 is behind a symmetric NAT, and SN_2 need to relay messages for UA_2. The SDP attributes of INVITE message, “c” and “m”, will be filled with SN_1 (IP address of SN_1) and Q_1 (allocated port in the SN for media sessions)

6. **REGISTER (hole punching for media sessions)**：UA_2 will also proceed the “hole punching for media sessions”. It sends REGISTER to its SN, SN_2.

7. **200 OK (hole punching for media sessions)**：SN_2 receives REGISTER message and fetches the “hole”. SN_2 records the “hole” and sends 200 OK response to UA_2.

8. **200 OK**：Then UA_2 returns 200 OK to UA_1. The SDP attributes, “c” and “m”, will be filled with SN_2 (IP address of SN_2) and Q_2 (allocated port on the SN for media sessions).

9. **ACK**：UA_1 receives 200 OK and returns ACK back.

10. Call setup procedure finishes and both UA_1 and UA_2 begin to talk and transfer media data. UA_2 will send media data to SN_1, and UA_1 will send media data to SN_2. These two SNs will relay media into the NAT in front of the UAs they are serving individually.

2-3-4. Solution 3’ (special case): Inward signaling/inward media/outward media passes through the super node

■ Special case of solution 3

The following figure shows a special case of solution 3, it occurs when a SIP UA behind a (port-) restricted cone NAT wants to call the other one behind a symmetric NAT or vice versa.

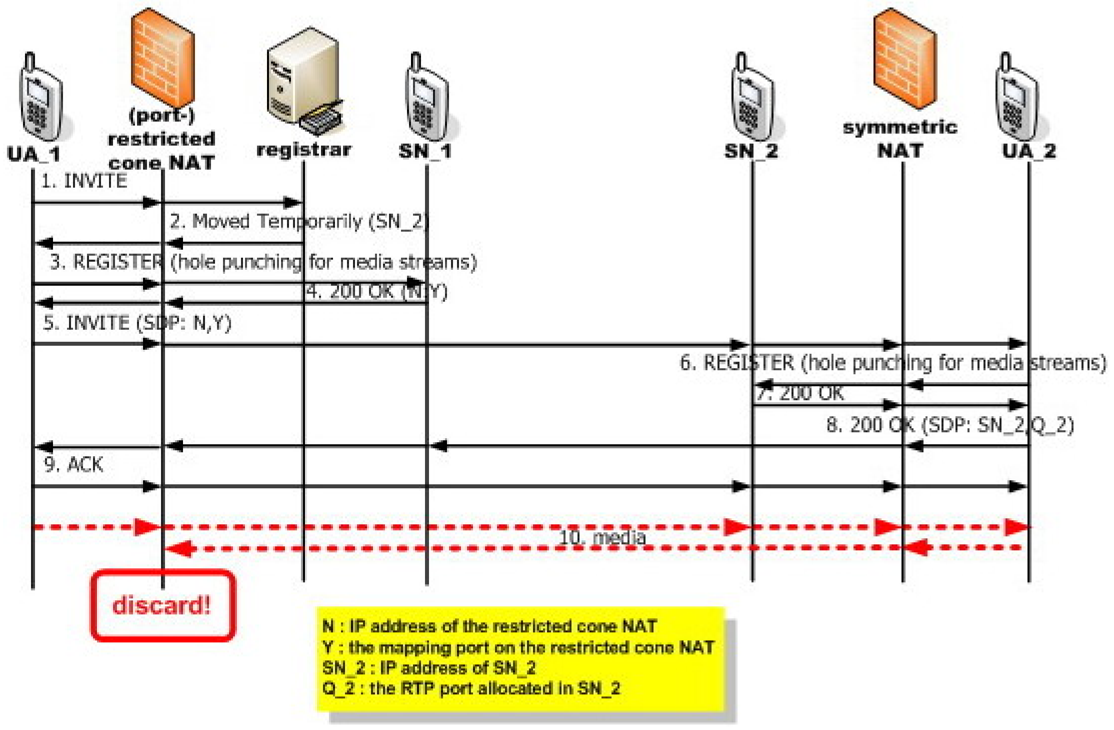

UA_1
(port-) restricted cone NAT
registrar
SN_1
SN_2
symmetric NAT
UA_2
1. INVITE
2. Moved Temporarily (SN_2)
3. REGISTER (hole punching for media streams)
4. 200 OK (N,Y)
5. INVITE (SDP: N,Y)
6. REGISTER (hole punching for media streams)
7. 200 OK
8. 200 OK (SDP: SN_2,Q_2)
9. ACK
10. media
discard!
N : IP address of the restricted cone NAT
Y : the mapping port on the restricted cone NAT
SN_2 : IP address of SN_2
Q_2 : the RTP port allocated in SN_2

Figure 2-8: special case of solution 3

Figure 2-9 describes such a special case: UA_1 is behind a port-restricted cone NAT; and UA_2 is behind a symmetric NAT. UA_1 wants to make a call with UA_2. The problem is that the RTP packets sending from UA_2 can’t pass through port-restricted cone NAT. When transmitting outward media steams, UA_1 punch a hole on port-restricted cone NAT between NAT and SN_2, but not UA_2 or symmetric NAT. Consequently, only SN_2 can send packets into port-restricted NAT and the RTP packets from UA_2 fails passing through port-restricted NAT according to the characteristic of port-restricted NAT:" an external host can send a packet, with source IP address X and source port P, to the internal host only if the internal host had previously sent a packet to IP address X and port P". To be distinguishable with solution 3, we call the solution "solution 3'"

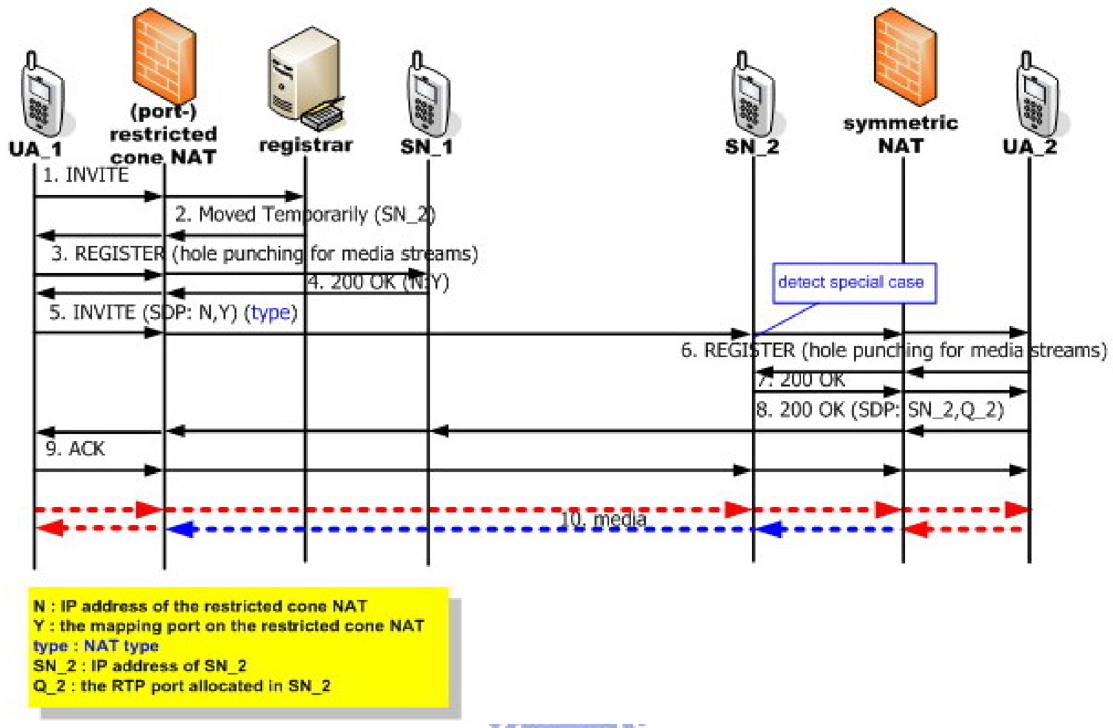

UA_1
(port-) restricted cone NAT
registrar
SN_1
SN_2
symmetric NAT
UA_2
1. INVITE
2. Moved Temporarily (SN_2)
3. REGISTER (hole punching for media streams)
4. 200 OK (N,Y)
5. INVITE (SDP: N,Y) (type)
detect special case
6. REGISTER (hole punching for media streams)
7. 200 OK
8. 200 OK (SDP: SN_2,Q_2)
9. ACK
10. media
N : IP address of the restricted cone NAT
Y : the mapping port on the restricted cone NAT
type : NAT type
SN_2 : IP address of SN_2
Q_2 : the RTP port allocated in SN_2

Figure 2-9: solution 3' example

1. **INVITE**：UA_1 sends INVITE message to its registrar. Each user who wants to setup a call with others will send INVITE to registrar first.

2. **302 Moved Temporarily**：Registrar receives the INVITE message, looks up its database for the location of the callee, and redirects the INVITE message to the SN of callee, SN_2.

3. **REGISTER (hole punching for media sessions)**：UA_1 will proceed the "hole punching for media sessions". It sends REGISTER to its SN, SN_1.

4. **200 OK (hole punching for media sessions)**：SN_1 receives REGISTER message and fetches the "hole" from socket. SN_1 records the "hole" and sends 200 OK response to

UA_1.

5. **INVITE**：UA_1 gets the location of the callee and sends INVITE to SN_2 because UA_2 is behind a symmetric NAT, and SN_2 need to relay messages for UA_2. Besides, the message must carry the type of the NAT .The SDP attributes of INVITE message, “c” and “m”, will be filled with N (IP address of NAT) and Y (mapping port on the NAT for media sessions). SN_2 receives REGISTER messages and fetches “type”. If the “type” is restricted cone, and the type of the NAT itself is symmetric, SN_2 will note that it is responsible for relay outward media sessions for UA_2 later and modify SDP attributes of the SIP message for UA_2. Then it relays REGISTER to UA_2.

6. **REGISTER (hole punching for media sessions)**： After receiving REGISTER, UA_2 will proceed the “hole punching for media sessions”. It sends REGISTER to its SN, SN_2.

7. **200 OK (hole punching for media sessions)**：SN_2 receives REGISTER message and fetches the “hole” from socket. SN_2 records the “hole” and sends 200 OK response to UA_2

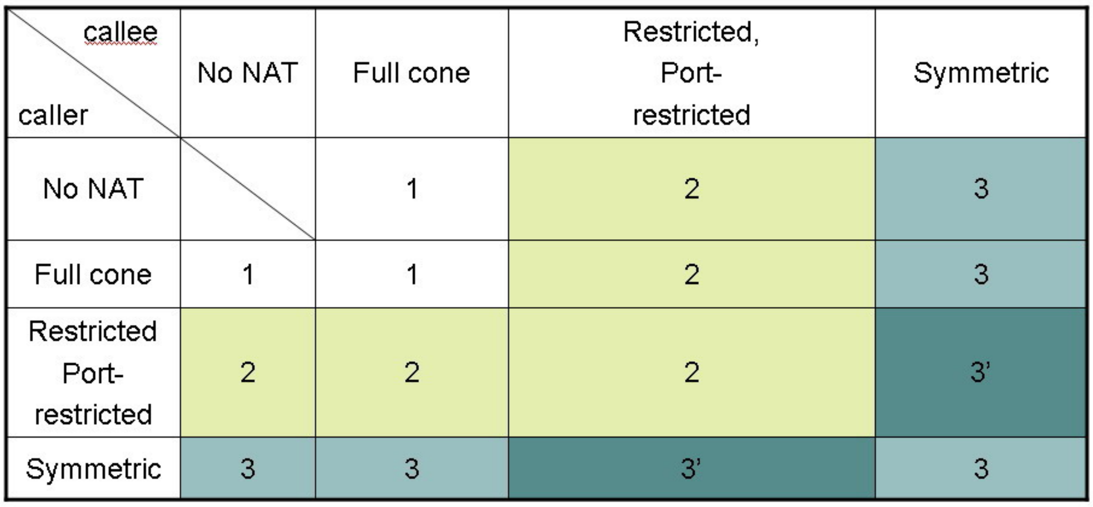
8. **200 OK**：Then, UA_2 returns 200 OK to UA_1. The SDP attributes, “c” and “m”, will be filled with SN_2 (IP address of SN_2) and Q_2 (allocated port in the SN for media sessions)

9. **ACK**：UA_1 receives 200 OK and returns ACK back.

10. Call setup procedure finishes and both UA_1 and UA_2 begin to transfer media data. UA_2 will send media data to its SN, SN_2, and UA_1 will send media data to SN_2, too. In other words, SN_2 will relay both inward and outward media for UA_2.

■ Call setup solution table

The following table describes the overall solutions for each kind of caller/callee case. (“1” means to the solution 1; “2” means to the solution 2; “3” means to the solution 3; and “3’” means to the solution 3’)



callee / caller	No NAT	Full cone	Restricted, Port-restricted	Symmetric
No NAT		1	2	3
Full cone	1	1	2	3
Restricted Port-restricted	2	2	2	3’
Symmetric	3	3	3’	3

Table 2-1: call setup solution table

■ Issues

There are still some issues in the thesis. First, we don’t consider the policy about choosing SN from SIP UAs in the public network. In the mechanism, a SN serves only one UA. There may be more advanced mechanism future for multiple SNs to serve the same UA. Second, we can design a failure tolerant methodology to recover the media when the serving SN fails or disconnects. Third, there may be more than one registrar in the network, so the negotiation between registrars seems significant for the manager.

Chapter 3 Design and Implementation

3-1. Implementation of NAT traversal

■ Hardware and software

We use CCL SIP UA as SIP UA in our experiments. The CCL SIP UA was implemented by the Computer & Communication Research Laboratories (CCL) of the Industrial Technology Research Institute. Registrar in the experiments is developed by Chi-Fan Lin, a member in our lab. We set up several NAT servers and verify their type, including D-Link DI 604 router (full cone NAT), EDIMAX wireless router (symmetric NAT), Linux Fedora Core release 3 (symmetric NAT), and Linux Mandrake 10 (symmetric NAT)

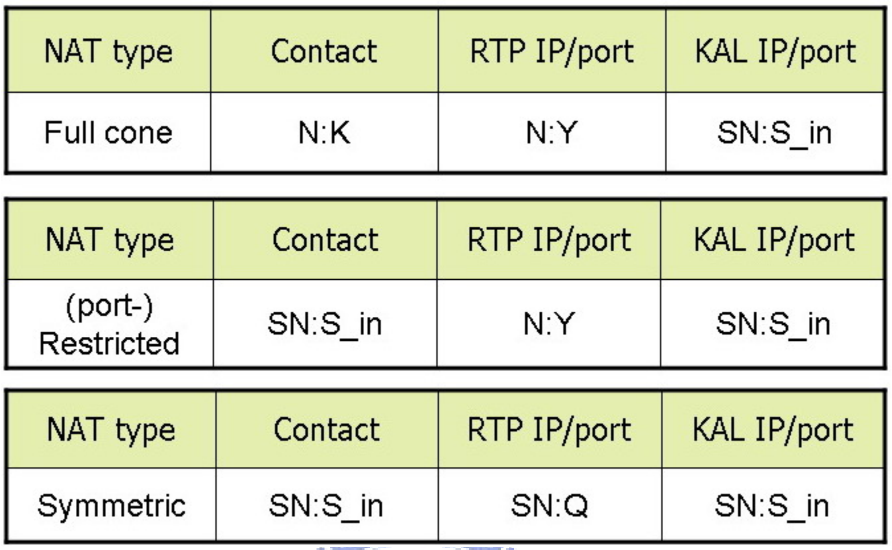
■ Data structure and protocol stack

Table 3-1: registrar table

ID	NAT type	Route IP/port
UA1	Full cone	NAT
UA2	(port-)Restricted	SN
UA3	Symmetric	SN
UA4	No NAT	UA

The table above is registrar table. A registrar receives registration from UAs, so it must record the ID of each UA. Based on the mechanism proposed in the previous chapter, registrar also has to record the NAT type for UA behind a NAT. The “Route” field means the location of each UA. A registrar will redirect each INVITE messages to the callee according to the address record in the “route” field. For a UA in the public network, the “Route” is the IP address itself; for an UA behind a full cone NAT, the “Route“ is the “hole” of the NAT, since for a full cone NAT, the packets from a external host can pass through NAT straightly ; for an UA behind a non full cone NAT, the "Route" is the IP/port of the SN which is responsible for serving it, since the packets must pass through SN in the solutions.

Table 3-2: UA (UA behind NAT) table



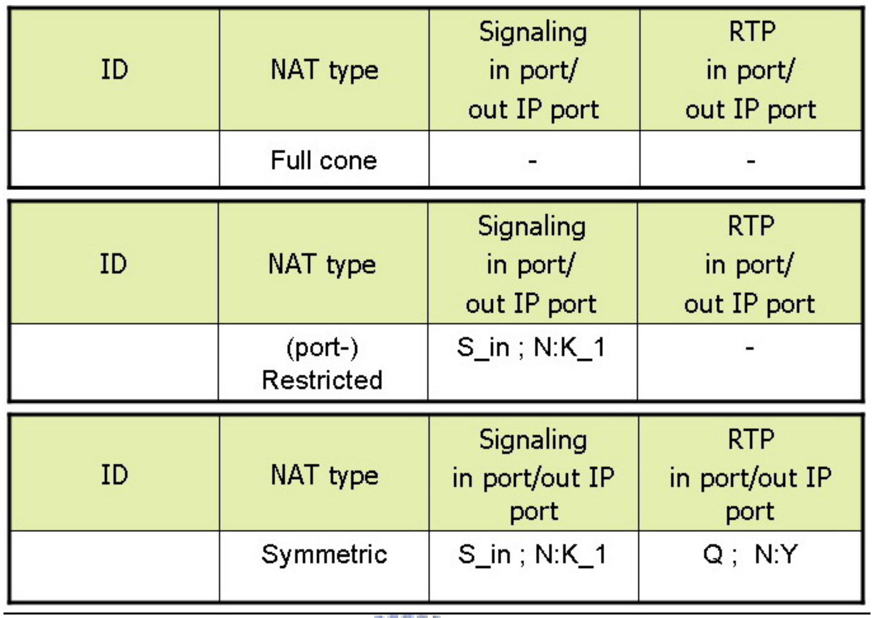
NAT type	Contact	RTP IP/port	KAL IP/port
Full cone	N:K	N:Y	SN:S_in

NAT type	Contact	RTP IP/port	KAL IP/port
(port-) Restricted	SN:S_in	N:Y	SN:S_in

NAT type	Contact	RTP IP/port	KAL IP/port
Symmetric	SN:S_in	SN:Q	SN:S_in

The table above is UA table maintained by the UA behind a NAT. UA behind a NAT must remember NAT type, "Contact" to modify "Contact" and "Via" header in SIP message so that the other peer have ideas where contact address to send response, "RTP IP/port" for SDP when making a call, and the IP address and port of its SN, which is record in the "KAL (keep-alive)" field in the table.

Table 3-3: UA (SN) table



ID	NAT type	Signaling in port/ out IP port	RTP in port/ out IP port
	Full cone	-	-

ID	NAT type	Signaling in port/ out IP port	RTP in port/ out IP port
	(port-) Restricted	S_in ; N:K_1	-

ID	NAT type	Signaling in port/out IP port	RTP in port/out IP port
	Symmetric	S_in ; N:K_1	Q ; N:Y

The table above is UA table maintained by SN. SN must record the ID of serving UA, NAT type to identify special case, the ports to relay signaling data, and the ports to relay media streams.

➢ **Protocol stack of SIP UA**

The top of the protocol stack is "UI" layer. The "UI" layer involves user interface, and reads user action to trigger corresponding "CallManager" functions. The "UAProfile" layer keeps all UA settings, such as local address, proxy, registrar…etc. The "CallManager" layer implements the core call model by providing the "UaCore" callback functions. It also refers to "MediaManager" layer for media control. The "MediaManager" layer refers to "WaveIO" layer for media playback, recording and refers to "cclRTP" layer for RTP handling. The "UaCore" layer is designed for UA kernel, and manages dialog, configuration objects. The "sipTx" layer is a single thread event-callback programming model. It supports four types of transaction defined in RFC 3261. The "SIP" layer implements the functions of SIP. The "transport" layer is responsible for low-layer API, such as socket management, and data structure…etc.

3-2. Demo scenarios

The following statements are the demo scenarios to verify our research. In the first case, we will verify the procedure of registration behind a full cone NAT. Second, we verify the registration behind a non full cone NAT. Third, we verify the call setup between a SIP UA in the public network and the other behind a full cone NAT. Forth, and we verify the call setup between a SIP UA in the public network and the other behind a symmetric NAT.

Chapter 4 Conclusions

Many service providers and private individuals use NAT to overcome the problem of not having enough IP addresses. However, NAT traversal problem somehow is troublesome and makes a SIP UA fail when one wants to call the other behind a NAT or vice versa. As a result, several solutions have been proposed. ICE is a quite good solution because it integrates STUN and TURN protocols and traverses all types of NATs. In this thesis, we presented a P2P-like solution that traverses all types of NATs with less message overhead than the ICE. In our solution, SIP UAs behind NAT determine the NAT type and assure an allocated address during SIP registration, and the registration is kept as simple as possible. Thus, they don't need to allocate redundant addresses and verify their connectivity when establishing calls. Moreover, the media "hole" will be punched until a call is about to be established to reduce the overhead of keep-alive messages. A SIP UA behind a NAT writes SDP attributes, such as "c=" and "m=", to indicated its NAT type, and the called party will send media data to the NAT or the SN of the internal UA in an optimal way.

In summary, the approach described in this thesis integrates most tasks including detection of NAT type, relay server assignment, hole punching with SIP registration. It also optimizes the procedure for UAs behind a NAT to initiate SIP calls. On the other hand, the policy about choosing SNs between users in the public network is also important. The next step is to find out an appropriate policy to enhance the efficiency of this architecture.

Reference

[1] P. Srisuresh, K. Egevang, “Traditional IP Network Address Translator (Traditional NAT)”, RFC 3022, January 2001.

[2] 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.

[3] K. Egevang, P. Francis, “The IP Network Address Translator (NAT)”, RFC 1631, May 1994

[4] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, E. Lear, “Address Allocation for Private Internets”, RFC 1918, February 1996

[5] J. Rosenberg, J. Weinberger, C. Huitema and R. Mahy, “STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)”, RFC 3489, March 2003.

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

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

[8] J. Rosenberg, R. Mahy and C. Huitema, “Traversal Using Relay NAT (TURN)”, draft-rosenberg-midcomturn- 03, October 2003.

[9] J. Rosenberg , “Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Multimedia Sessions Establishment Protocols”, draft-ietf-mmusic-ice-04, February 2005

[10] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, Scott Shenker “A Scalable Content-Addressable Network”, In Proc. ACM SIGCOMM, San Diego, CA, August 2001

[11] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan “Chord:

A Scalable Peer-to-peer Lookup Service for Internet Applications", In Proc. ACM SIGCOMM, San Diego, CA, August 2001

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

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