How NAT-Compatible Are VoIP Applications?

Ying-Dar Lin, Chien-Chao Tseng, Cheng-Yuan Ho, and Yu-Hsien Wu, National Chiao Tung University

ABSTRACT

Today, the Internet is very diversified, further complicating the classic NAT traversal problems. In order to solve these problems, there are many proposed methods classified into two categories. One enhances the NAT traversal techniques of applications, and the other tries to modify the behavior of NATs. In this article we focus on the former because NATs have been installed, and their behavior cannot be altered through endpoint users. Accordingly, in order to test NAT traversal techniques of five VoIP applications (Skype, MSN, Google Talk, X-Lite, and Linphone), three network topologies have been designed with two endpoints behind the same, different, or multilevel NATs. Through a series of experiments and from the experiment results, we observe that these VoIP applications use some traversal techniques, such as NAT mapped-address probe, peer discovery, path check, and relay first, proposed by STUN, TURN, and ICE to make a direct connection when endpoints are behind the same or different NATs with independent mapping rules. However, with multilevel NATs, no endpoints can establish a direct connection when they use the above mentioned techniques, even if hairpin behavior is supported by NATs.

INTRODUCTION

Peer-to-peer (P2P) applications, such as BitTorrent and Skype, have gained in popularity in recent years. The Internet environment is, as of now, still based on IPv4 systems, so not every computer is awarded its own public IP address due to the address shortage. Therefore, IPv6 and Network Address Translation (NAT)[1] [1] over IPv4 were proposed. IPv6 promises great usable address space while NAT allows computers in private networks to access resources in public networks.

The NAT techniques gain more widespread deployment than IPv6. However, they introduce NAT traversal problems. An internal node (IN) behind an NAT router cannot be contacted by an external node (EN) unless the IN initiates the communication first. Furthermore, it becomes worse that two nodes cannot directly communicate with each other when both nodes are behind different NATs. The NAT traversal problem is less likely to occur in traditional client-server applications, where servers are publicly accessible, but common in P2P applications. Thus, how to let an EN or IN communicate with an IN (i.e., the NAT traversal mechanism) is important and necessary in today's Internet [2, 3].

In order to operate well behind NATs, VoIP applications use two different ways to make a connection. One is through direct peer-to-peer connections, and the other is through a machine with a public IP (i.e., a third party). Although the latter allows VoIP applications to work behind NATs, it may lead to other problems like high bandwidth cost, server overloading, and packet delay, which are especially harmful to real-time P2P applications such as voice over IP (VoIP) [4]. Accordingly, in this article we pay more attention to the ability of VoIP applications establishing *direct* peer-to-peer connections through NATs than the *relayed* peer-to-peer connections.

Several NAT traversal protocols have been proposed, such as Session Traversal Utilities for NAT (STUN) [5–7], Traversal Using Relays around NAT (TURN) [8], and Interactive Connectivity Establishment (ICE) [9]. STUN is a protocol an IN can use to detect its NAT type and mapped-address (i.e., external global IP address and port number), while TURN assists an IN and a corresponding EN in relaying packets; ICE, on the other hand, makes use of both STUN and TURN to check possible connection between two nodes. If an IN wishes to establish a connection, it must initiate the connection by itself or notify an EN about its NAT mapped-address information. Furthermore, before a connection is made, STUN helps VoIP applications to get their NAT mapped-address, and mapping and filtering rules, explained in the next section. In order to solve the problem that an EN cannot connect to an IN on its own initiative, TURN uses a relay server, which is contacted by the EN and the IN, to relay all packets between them. ICE makes use of both STUN and TURN with a path check algorithm, which can establish a connection, either a direct peer-to-peer connection or through a relay, between two nodes.

Therefore, in this article we aim to discuss whether five VoIP applications establish direct peer-to-peer connections behind NATs in some typical network topologies, and whether VoIP applications have developed their own *NAT*

[1] *Some users may consider an NAT as a firewall because both restrict communication between a private network and the public Internet. However, the NAT modifies the header information, such as IP address and TCP/UDP port, in packets flowing across the boundary while a firewall does not change any header information in packets. Furthermore, one can configure the firewall's access control list (ACL) to filter the unwanted traffic, but cannot do this to the NAT devices.*

0163-6804/10/$25.00 © 2010 IEEE

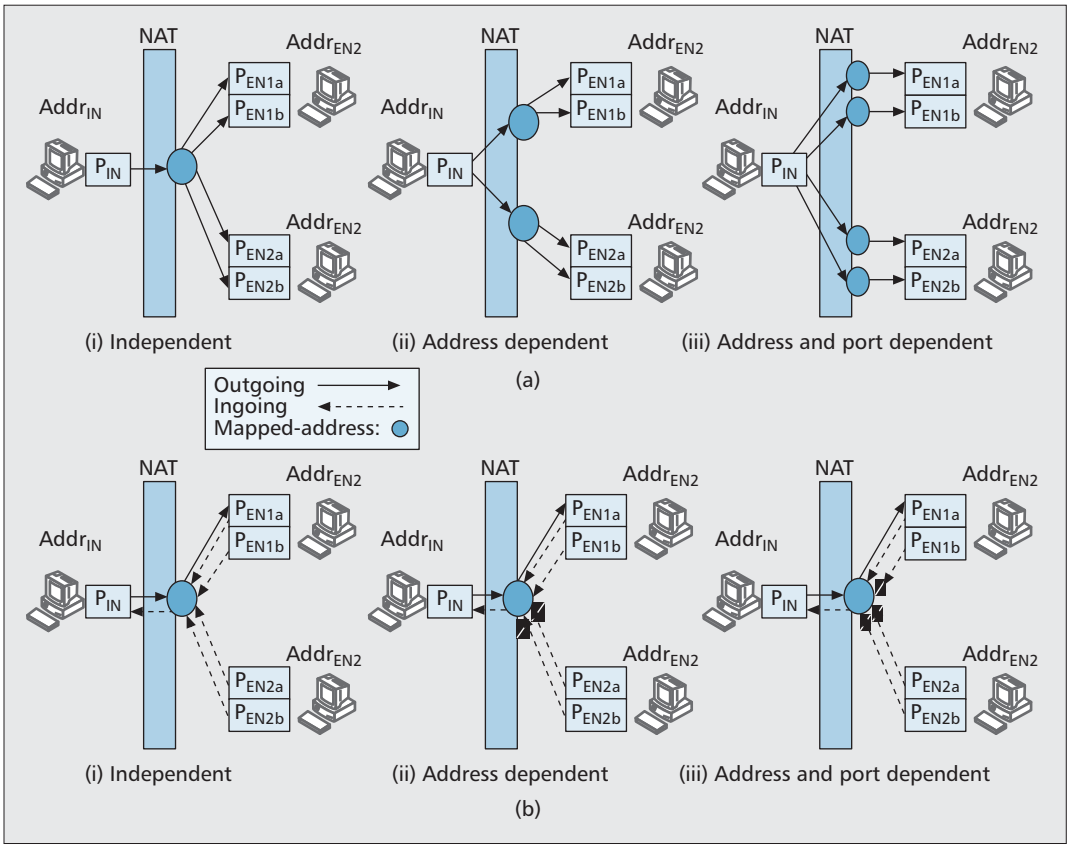

Figure 1. *Mapping and filtering rules: a) mapping; b) filtering.*

In order to operate well behind NATs, VoIP applications use two different ways to make a connection. One is through direct peer-to-peer connections, and the other is through a machine with a public IP, i.e., a third party.

traversal technologies (NTTs) or shared common behaviors among them. Through a series of real experiment tests and result analyses, we shall observe whether these VoIP applications use peer-to-peer connections when the caller and callee are behind the *same* NAT. On the other hand, while the caller and callee are behind *different* NATs, which VoIP applications are able to use direct peer-to-peer while the others have to use relays to establish connections? This could be related to their NTTs. Through further analyses, we shall categorize five common NTTs of VoIP applications, in the order of general use, NAT mapped address probe, peer discovery, path check, port prediction, and relay first. These technologies shall be presented in detail later.

The remainder of this article is organized as follows. We first describe mapping and filtering rules of an NAT in detail, and then introduce three NAT traversal protocols and discuss their advantages and drawbacks. In the first half of the following section, we describe the setup and design of experiments and network topologies in detail; in the second half, we analyze the experiment results, and discuss whether the NTTs we observed influence the tests or not. Finally, we summarize our findings and provide suggestions for further research in the final section.

Literature Review

Network Address Translation

According to [10], a *mapped-address* is the external global IP address and port number allocated by an NAT for a connection attempt establishing from an IN, while a *filtering* shows how the NAT handles (or discards) packets sent by an EN trying to use an existing mapped-address. The rules and classifications of mapped-address allocation, also called *mapping*, and filtering are described as follows [5]:

- Mapping: How an NAT allocates a mapped-address to an IN's outgoing connections:
  - –Independent: All requests from the same internal IP address and port, to *any* destination IP address or port, are mapped to the same external global IP address and port (Fig. 1a.i).
  - –Address dependent: All requests from the same internal IP address and port, to the *same destination IP address*, are mapped to the same external global IP address and port (Fig. 1a.ii).
  - –Address and port dependent: All requests from the same internal IP address and port, to a *specific destination IP address and port*, are mapped to a unique external IP address and port (Fig. 1a.iii).
- Filtering: How an NAT handles an EN's ingoing packets to an existing mapped-address:
  - –Independent: *Any* EN can send a packet to the IN, by sending a packet to the mapped-address (Fig. 1b.i).
  - –Address dependent: An EN (with an IP address Addr$_{EN}$) can send a packet to the IN only if the IN had previously sent a packet to the IP address Addr$_{EN}$ of the EN (Fig. 1b.ii).

STUN is a protocol that allows applications to discover the presence and rules of NATs between them and the public Internet. It also provides the ability for applications to determine the external global addresses allocated to them by NATs.

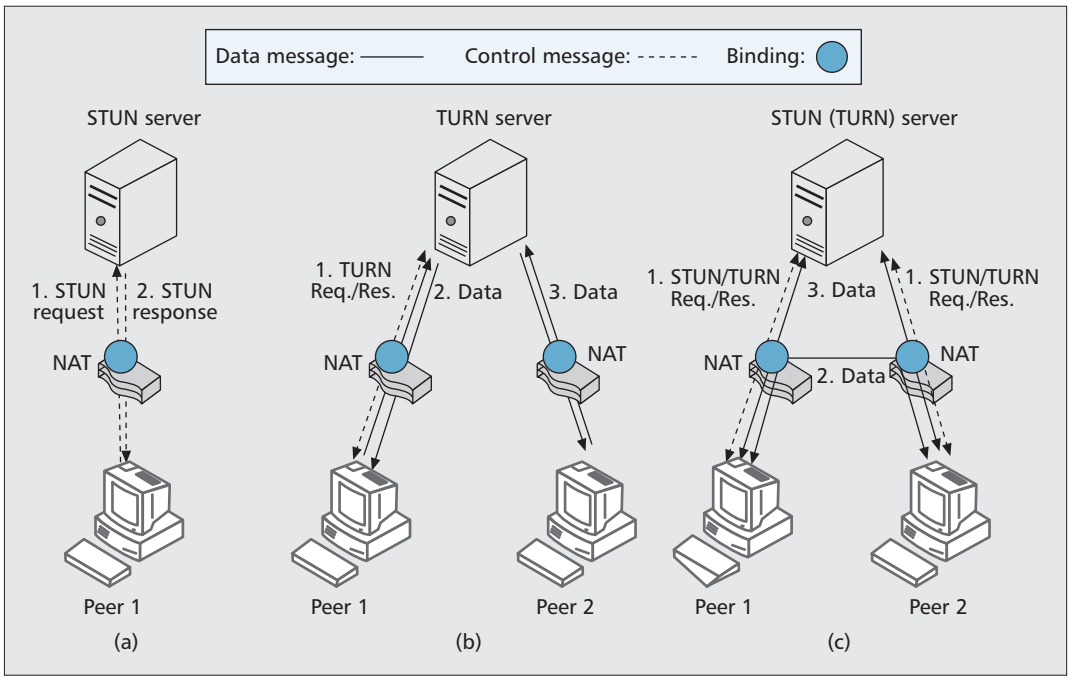

Figure 2. *Common NAT Traversal protocols for P2P applications: a) STUN; b) TURN; c) ICE.*

–Address and port dependent: An EN can send a packet, with a source IP address $Addr_{EN}$ and a source port P_{EN}, to the IN only if the IN had previously sent a packet to the IP address $Addr_{EN}$ and port P_{EN} (Fig. 1b.iii).

A *hairpin* translation (or called *loopback* translation) is a technology defined in [2, 3]. The definition of hairpin is as follows. Suppose that two peers, IN1 and IN2, are behind the same NAT and exchanging packets via the mapped external addresses of each other. An NAT is said to support hairpin if it can relay packets for IN1 and IN2. Furthermore, not all contemporary NATs support hairpin.

STUN, TURN, AND ICE

Many solutions have been proposed to solve P2P NAT traversal problems. In this subsection we introduce and compare three commonly used protocols: STUN, TURN, and ICE.

STUN is a protocol that allows applications to discover the presence and rules of NATs between them and the public Internet. It also provides the ability for applications to determine the external global addresses allocated to them by NATs. For example, an IN sends a request to the STUN server and gets its external NAT mapping and filtering rules, and mapped-address from the response of the STUN server as shown in Fig. 2a. How the IN passes this information to ENs to initiate connections and transmits data packets depends on applications. STUN works with many existing NATs, and does not require any special behavior of them. As a result, it allows a wide variety of applications to work with existing NAT infrastructure. Moreover, STUN may help incoming packets to pass through NAT devices with the independent filtering rule, but not through NAT devices with the address (and port) dependent filtering rule [5–9]. This is because an EN can send a packet, with a source IP address $Addr_{EN}$ (and a source port P_{EN}), to the IN only if the IN had previously sent a packet to the IP address $Addr_{EN}$ (and port P_{EN}).

TURN is a protocol that allows the host to communicate with relays so that it can exchange packets with its peers using relays and resolves the NAT traversal problem which cannot be solved by STUN. Furthermore, TURN differs from some other relay control protocols in that it allows a client to communicate with multiple peers using a single relay address. As presented in Fig. 2b, an IN behind an NAT must first send an allocation request to a relay server to request a relay address, which consists of an IP address and a port number. Once the relay server has allocated a relay address for the IN, it will start forwarding all packets destined for the relay address to the IN. Therefore, other nodes that wish to send packets to the IN can then use the relay address to communicate with the IN through the relay server, and thus resolve the traversal problem of NATs with the address and port dependent filtering rule.

The Internet Engineering Task Force (IETF) proposed ICE as a protocol for NAT traversal for UDP-based media streams (although ICE can be extended to handle other transport protocols, e.g., TCP) established by the offer/answer model. As shown in Fig. 2c, a peer first acquires its external NAT mapped address from a STUN server, and a relay address from a TURN server. Then it uses some out-of-band protocol such as Session Initiation Protocol (SIP) to notify the other peer of its mapped and relay addresses. The two peers then use ICE to do the connectivity check for each possible path. If the two peers can establish a direct connection successfully, the two peers will use this connection to transmit their packets like message 2 of Fig. 2c. Otherwise, the two peers will exchange their packets via the relay server as presented in message 3 of

	STUN	TURN	ICE
Main functions	Detect NAT's mapping and filtering rules, mapped-address	Support a relay	Do connectivity checks
Traverse NAT with the independent filtering rule	O	O	O
Traverse NATs with the address/address and port dependent filtering rules	X	O	O
Required server	STUN server	TURN server	STUN and TURN server
Communication protocol	UDP/TCP	UDP/TCP	UDP/TCP

(O: Yes, support; X: No, nonsupport)

Table 1. *Comparison on existing NAT traversal techniques.*

Fig. 2c. Table 1 summarizes the main functions of STUN, TURN and ICE, whether they have abilities to traverse different kinds of NATs, what service machines they need, and what protocols they could support.

In the real Internet, there are several types of attacks possible in STUN, TURN, and ICE deployments. For example, an attacker may modify STUN, TURN, or ICE messages in transit; may observe STUN, TURN, or ICE requests, and then immediately send a response, typically an error response, to disrupt STUN, TURN, or ICE processing; may compromise the DNS, causing DNS queries to return a rogue STUN and TURN server addresses; and so on. The countermeasures of STUN, TURN, and ICE include coworking with secure signaling techniques such as SIPS, using authentication and encryption methods, limiting the resource to abnormal processes, and so on. Because of limited space, more detail can be found in [5, 8, 9].

Technically, STUN provides NAT information to help peers to establish direct connections, and TURN relays packets to help two peers communicate with each other. In short, from a *traverse-NAT* point of view, the success rate of TURN is 100 percent because even when both endpoints are behind NATs with the address and port dependent filtering rule, they can still communicate with each other through a TURN server. However, using TURN servers may cause longer delay, which would affect media quality and also consume more bandwidth because the same packets pass through the server twice. Therefore, direct connections should be used if possible. This is why ICE gives higher priority to a direct connection. Accordingly, all experiments in this article focus on the exploration of *whether or not peers behind NAT can connect to others directly*.

Experiment Design and Results

The network topology of today's Internet is extremely diversified, which may make NAT traversal problems more complicated. Will current NAT traversal technologies not work because of network environment changes? Could VoIP peers behind different NAT devices use direct peer-to-peer connections? To understand the NTTs of VoIP applications and how the caller and callee, behind different NAT network topologies, establish communication using direct connections or relaying through an external host, a series of experiments have been designed. From the results, how the caller and callee establish connections and the direct connection rate (DCR) are summarized.[2]

Experiment Environment Setup

In order to know how NAT devices affect VoIP connections, eight representative NAT devices, whose detailed information is shown in Table 2, and five VoIP applications — Google Talk, MSN, Skype, X-Lite, and Linphone — are chosen.[3] These five VoIP applications may not directly use SIP, STUN, TURN, or ICE, but the essences of the technologies they use are similar to that of SIP, STUN, TURN, and ICE. Therefore, in the following paragraphs we use SIP, STUN, TURN, and ICE to represent the respective technologies used by the five VoIP applications. Furthermore, on the official website, from release notes and updated information of each chosen VoIP application version, there are no changes of NTTs, but changes in the user interface (UI) and codecs over the last revision.

To establish a more realistic Internet environment, in Fig. 3 we design three different kinds of network topologies. In each network topology there are two hosts (a caller and a callee) and a SIP server. The two hosts execute the same VoIP application to make a phone call. Moreover, two hosts may be behind the same NAT as shown in Fig. 3a, or behind different NATs as shown in Figs. 3b and 3c. As for the SIP server and STUN/TURN server, they are located in a public IP domain to provide services like registration, NAT mapped-address notification, and relay if VoIP applications require. For example, all VoIP applications have to register to the SIP server, but do not use the TURN server to relay packets when they can establish a direct connection.

However, in a network topology as shown in Fig. 3c, if the uppermost layer of NAT does not support hairpin translation, packets cannot go directly from a lower-layer NAT through this one to reach another lower-layer NAT. This causes inability for the two peers to connect directly. Thus, in this kind of network topology, to enable direct connections of VoIP applications, an NAT that supports hairpin

[2] *When VoIP service providers construct their network topology, they can refer to these results to consider how many STUN and TURN servers are needed.*

[3] *From the Software Informer website, http://software.informer.com, the top ten popular VoIP software are MSN, Skype, Yahoo! Messenger, Windows Live Call, Google Talk, Ventrilo, X-Lite, SmartVoip, Jumblo, and Vbuzzer Messenger. MSN, Skype, Google Talk, and X-Lite are chosen because they can execute both on Windows and Linux. To compare their NTTs, Linphone, a simple VoIP application, is chosen to be the control group.*

Brand	Model	Firmware	Mapping rule	Filtering rule	Hairpin
SMC	Smcwbr14-g2	V1.05	Independent	Independent	Yes
D-Link	Di-604	V3.14(tw)	Independent	Independent	Yes
Windows	Ics	Xp-sp2	Independent	Independent	No
Linksys	Befsr41	2.00.02	Independent	Address and port dependent	No
3Com	3crwer 100-75	1.2.0	Independent	Address and port dependent	No
Linux	iptables	1.3.5	Independent	Address and port dependent	No
Netgear	Pr614	V0.1.8_03.17	Address and port dependent	Address and port dependent	No
FreeBSD	Pf	6.2 release	Address and port dependent	Address and port dependent	No

Table 2. *Required NAT devices in the experiments.*

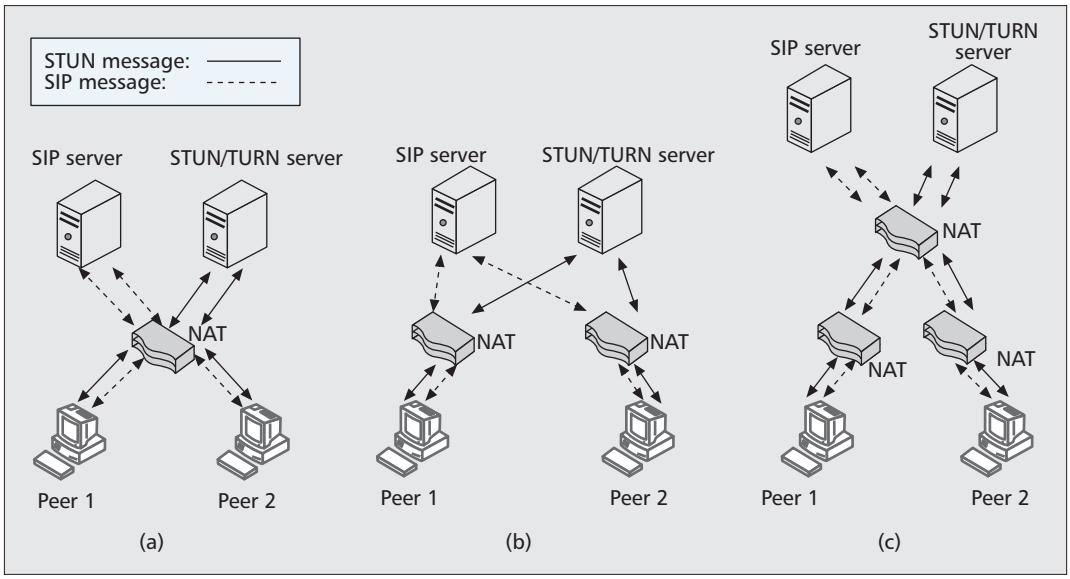

Figure 3. *a) Peers behind a common NAT router; b) peers behind different NAT routers; c) peers behind multiple levels of NAT routers.*

translation must be chosen for the uppermost layer; otherwise, any packets sent on any direct connection paths will be filtered and disregarded by the NAT device. After these eight NAT devices are tested by the stund [2] program, it is discovered that only D-Link and SMC NAT devices support hairpin translation. Therefore, one of the two must be used as the uppermost layer of NAT in this kind of network topology.

Because there are eight NAT devices, for any VoIP applications, either eight,

$$\binom{8+2-1}{2} = 36,$$

or

$$\binom{8+2-1}{2} * 2 = 72$$

NAT combination tests, shown in Figs. 3a, 3b, and 3c, respectively (i.e., totaling 116 tests), are performed. During each test, all procedures and packet flows are recorded for analysis. The steps for each test are as follows:

Step 1. Two peers execute the same VoIP application.

Step 2. In the application's functional setup, we enable all capabilities relevant to NAT traversal in order to understand whether this application can traverse a NAT device. Furthermore, the IP addresses of the STUN/TURN server are filled, if necessary. For example, we can add the proxy information to MSN's and Skype's connection setting field if needed.

Step 3. After registering to the SIP server, two peers start to establish a call. The caller dials, and the callee accepts this invitation as soon as the signal is received. This call keeps alive for 30 s.

Step 4. After 30 s, the caller not only ends the call but also shuts down the application.

(a)

	Google Talk	MSN	Skype	X-Lite	Linphone
NAT mapped-address probe (STUN)	Yes	Yes	Yes	Yes	Yes
Peer discovery (ICE)	Yes	Yes	Yes	Yes	No
Path check (ICE)	Yes	No	Yes	No	No
Port prediction	No	No	Yes	No	No
Relay first (TURN)	No	Yes	No	No	No

(b)

	Google Talk	MSN	Skype	X-Lite	Linphone
Topology 1 (Fig. 2a)	8/8 (100%)	8/8 (100%)	8/8 (100%)	8/8 (100%)	2/8 (25%)
Topology 2 (Fig. 2b)	26/36 (72%)	0/36 (0%)	28/36 (78%)	23/36 (64%)	10/36 (28%)
Topology 3 (Fig. 2c)	0/72 (0%)	0/72 (0%)	0/72 (0%)	0/72 (0%)	0/72 (0%)

Table 3. *Observed NTTs and DCR of 5 VoIP applications: a) observed NTTs of 5 VoIP applications; b) the DCR of 5 VoIP applications with different NAT combinations in three different network topologies.*

Step 5. Two peers stop recording packet flows and save to a separate pcap file.

In order to verify how two peers communicate with each other, we trace the pcap file to understand the packet flows between two peers. If two peers receive each other's RTP packets with the source IP address and port number being the IP address and port number of each other's NAT device (i.e., NAT mapped-address), this connection is direct. If not, this means that packets are relayed by an external host.

Experiment Results, Analysis, and Discussion

VoIP Application Behaviors — In this series of experiments an NAT is seen as a black box. From the record of packet flows, we observe that each VoIP application has different NTTs and also define five NTTs. In order of generality, the name and definition of five NTTs are as follows:

1. NAT mapped-address probe [5, 6]: Before making a connection, a client gets its NAT mapped-address from a STUN server.
2. Peer discovery [3]: A caller/callee tests a callee/caller to see if they are behind the same NAT.
3. Path check: A series of tests are done to confirm whether a direct or relay path exists between two peers.
4. Port prediction [2]: When a caller/callee is behind an NAT with an address and port dependent mapping rule, a callee/caller predicts and sends a test packet to a mapped-address that this NAT may allocate to the caller/callee. If the callee/caller can receive the response of the test packet from the caller/callee, it has predicted the NAT mapped-address correctly. Moreover, in order to increase the probability of prediction, a VoIP peer (e.g., the callee/caller in the above example) will send many packets by increasing destination port. The reason for increasing destination port linearly to estimate the NAT mapped-address is that an NAT with an address and port dependent mapping rule either linearly or randomly generates a mapped-address, and the linear increase is straightforward to approach the mapped-address.
5. Relay first: Before establishing a direct connection, two peers transmit data through a relay. This reduces delay caused by discovering a direct connection.

Table 3a shows above five NTTs with the order of general use for traversing the NAT and the comparison of five VoIP applications. Table 3b presents the DCR of 5 VoIP applications with different NAT combinations in three different network topologies. From Table 3a, we first observe that all VoIP networks have an NTT NAT mapped-address probe, which is provided by STUN. This means a VoIP user get its NAT mapped-address is basic for NAT traversal. Although both peer discovery and path check are provided by ICE, the former is popularly used by most VoIP applications (i.e., only Linphone does not have it). On the other hand, path check is implemented in Google Talk and Skype. The reason for this phenomenon may be that path check has higher latency than peer discovery. Finally, only Skype uses port prediction, and MSN supports relay first, which is provided by TURN. This may be because the probability of guessing the correct mapped-address by port prediction is very low and the company and corporation of the VoIP application needs to set up

we may say that the DCR of current VoIP applications behind different NATs still has room for improvement. In particular, the open source VoIP application, compared with other VoIP applications, has a very poor DCR.

a TURN server or a relay server if using relay first. In these five VoIP applications, Skype has the most NTTs such that it may traverse the most NAT devices.

Most VoIP Applications can Traverse the Same NAT Device under Topology 1 — From Table 3b, we can see that under the first network topology (Fig. 3a), Linphone's DCR is the lowest at 25 percent, while all the other VoIP applications reach 100 percent. This is because Linphone does not have peer discovery. Accordingly, under the first network topology, a Linphone peer cannot check whether other peers are behind the same NAT as it is. Furthermore, the reason Linphone's DCR is not zero is that D-Link and SMC NATs support hairpin translation.

Some VoIP Applications Can Traverse Certain NAT Devices under Topology 2 — From Table 3b, we can see that under the second topology (Fig. 3b), Skype's DCR is the highest of all VoIP applications at 78 percent, Google Talk's is the second highest at 72 percent, and the lowest one is MSN at 0 percent. Skype attains the limit of DCR in this topology because, according to [11], it is impossible for applications to establish a direct connection when one peer is behind an NAT with address and port dependent mapping and filtering rules, and the other peer is behind an NAT with address and port dependent filtering rules. On the other hand, in other NAT combinations, Skype should establish a direct connection easily. Therefore, in this topology, the possible upper bound is the DCR of Skype, 78 percent.

Furthermore, compared with Table 3a, we can discover that Skype and Google Talk only differs in *port prediction*, but this was enough to result in a 6 percent difference in making direct connections. Therefore, we may derive that the ability port prediction is helpful for traversing NAT under topology 2. As for the ability to do an *NAT mapped-address probe*, this may have become a standard ability because all chosen VoIP applications have it. Finally, the DCR of MSN may have been zero because MSN always uses relay to transmit its packets except when two users are in the same subnet. This causes MSN not to execute another NTT under topology 2; however, under topology 1, it executes peer discovery after it directly receives packets from the other peer. This is why MSN's caller and callee establish a direct connection behind the same NAT.

No VoIP Application Can Traverse Multilevel NAT Devices under Topology 3 — From Table 3b, we can see that none of the abilities in Table 3a can help two peers establish a direct connection with each otherunder topology 3 (Fig. 3c) even if the uppermost NAT supports hairpin translation. From the record of packet flows, we observe that the uppermost layer of NAT cannot loopback packets in multilevel NATs. We cannot know the realistic reason since the NAT is seen as a black box. However, we think the test flow of the stund program and that of this experiment are different because the former uses one peer to send packets from its one port to the NAT mapped-address created by its other port behind an NAT, and the latter uses two peers to establish a direct connection in multilevel NATs. Therefore, the hairpin translation support result of the stund program may not be suitable for this experiment.

Conclusions

NAT-Compatible VoIP Applications?

In this article a series of experiments are conducted on five VoIP applications to test their NTTs under 116 combinations of eight NAT devices and three network topologies. From experiment results, we observe that:

- When two peers are behind the same NAT device, most VoIP applications can let peers connect to each other directly.
- When two peers are behind different NAT devices, only some VoIP applications in combination with certain NAT devices could make a direct connection.
- When two peers are behind two layers (or multiple layers) of NAT devices, none of the VoIP applications can pass through NATs to establish a direct connection.

Thus, we may say that the DCR of current VoIP applications behind different NATs still has room for improvement. In particular, the open source VoIP application, compared with other VoIP applications, has a very poor DCR.

Observed VoIP Application Behaviors

From the experiment results, we observe that VoIP applications could have five NTTs. In the order of generality, they are NAT mapped-address probe, peer discovery, path check, port prediction, and relay first.

NAT mapped-address probe helps a VoIP user get the mapped-address and thus solves many direct connection problems in a network topology with an NAT with independent mapping rule such that it is basic to the NAT traversal techniques. Peer discovery may reduce the time it takes for VoIP applications to establish direct connections behind the same NAT device so many VoIP applications implement it. Path check and port prediction increase the DCR because they help peers to find a direct connection behind different NATs. This is because path check does a series of tests to confirm whether a direct or relay path exists between two peers and port prediction can estimate changes in mapped addresses caused by NATs with address and port dependent mapping rule. Relay first reduces delay caused by discovering a direct connection. However, path check may have higher latency, the probability of guessing the correct mapped address by port prediction is very low, and relay first needs to set up a TURN server or a relay server. Therefore, these three NTTs are rarely implemented in VoIP applications.

Relations between Three NAT Traversal Protocols and VoIP Applications

STUN has NAT mapped-address probe ability, and TURN can provide relay first ability. ICE not only has peer discovery and path check abilities, but also makes use of both STUN and TURN.

From the experiment results, we observe that STUN is implemented in all VoIP applications, TURN is only used by MSN, the peer discovery of ICE is not developed in Linphone only, and the path check of ICE is embedded into Google Talk and Skype. Besides, Skype has its own port prediction technology. Consequently, Skype has the highest DCR among five VoIP applications.

Suggested Evolution of NATs

In network topologies like Figs. 2b and 2c, Skype peers still cannot establish a direct connection when one peer is behind an NAT with address and port dependent mapping and filtering rules, and the other peer is behind an NAT with address and port dependent filtering rules, or both peers are behind multilevel NATs. This is because there is no satisfactory resolution for VoIP applications to establish a direct connection. In other words, the current existing NTTs cannot traverse these NAT combinations. Accordingly, the possible upper bound of DCR is the DCR of Skype, and there is no space to enhance NTTs if we cannot change the NAT infrastructures. On the other hand, we suggest more NAT infrastructure using independent mapping rules, making VoIP more NAT-compatible if the NAT infrastructure can be varied.

References

[1] K. Egevang and P. Francis, "The IP Network Address Translator (NAT)," IETF RFC 1631, May 1994.
[2] B. Ford, P. Srisuresh, and D. Kegel, "Peer-to-Peer Communication Across Network Address Translators," *Proc. USENIX Annual Tech. Conf.*, Anaheim, CA, Apr. 2005, pp. 179–92.
[3] P. Srisuresh, B. Ford, and D. Kegel, "State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs)," IETF RFC 5128, Mar. 2008.
[4] H. Khlifi, J.-C. Gregoire, and J. Phillips, "VoIP and NAT/Firewalls: Issues, Traversal Techniques, and a Real-World Solution," *IEEE Commun. Mag.*, vol. 44, no. 7, July 2006, pp. 93–99.
[5] J. Rosenberg et al., "Session Traversal Utilities for NAT (STUN)," IETF RFC 5389, Oct. 2008.
[6] R. Denis-Courmont, "Test Vectors for Session Traversal Utilities for NAT (STUN)," IETF RFC 5769, Apr. 2010.
[7] D. MacDonald and B. Lowekamp, "NAT Behavior Discovery Using Session Traversal Utilities for NAT (STUN)," IETF RFC 5780, May 2010.
[8] J. Rosenberg, R. Mahy, and P. Matthews, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)," IETF RFC 5766, Apr. 2010.
[9] J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols," IETF RFC 5245, Apr. 2010.
[10] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," IETF RFC 3022, Jan. 2001.
[11] Y. Wang, Z. Lu, and J. Gu, "Research on Symmetric NAT Traversal in P2P Applications," *Proc. ICCGI*, 2006, p. 59.

Biographies

Ying-Dar Lin (ydlin@cs.nctu.edu.tw) is a professor in the Department of Computer Science at National Chiao Tung University, Hsinchu, Taiwan. He is the founder and director of the Network Benchmarking Laboratory (NBL; www.nbl.org.tw) since 2002. He received a Ph.D. in computer science from the University of California, Los Angeles in 1993. He is currently on the editorial boards of *IEEE Communications Letters*, *IEEE Communications Magazine*, *IEEE Communications Surveys & Tutorials*, *Computer Networks*, and *Computer Communications*.

Chien-Chao Tseng (cctseng@cs.nctu.edu.tw) is currently a professor in the Department of Computer Science at National Chiao Tung University. He received his B.S. degree in industrial engineering from National Tsing-Hua University, Hsinchu, Taiwan, in 1981, and his M.S. and Ph.D. degrees in computer science from Southern Methodist University, Dallas, Texas, in 1986 and 1989, respectively. His research interests include wireless Internet, handover techniques for heterogeneous networks, and mobile computing.

Cheng-Yuan Ho (cyho@csie.nctu.edu.tw) is an assistant research fellow in the D-Link NCTU Joint Research Center and the Information and Communications Technology Laboratory of the Microelectronics and Information Systems Research Center at National Chiao Tung University. His research interests include design, analysis, and modeling of the congestion control algorithms, real-flow test, high-speed networking, embedded hardware-software co-design, quality of service, and mobile and wireless networks. Ho received a Ph.D. in computer science from National Chiao Tung University in 2008.

Yu-Hsien Wu (yswu.cs96g@g2.nctu.edu.tw) received an M.S. degree in computer science from National Chiao Tung University in 2009. His research interests include high-speed networking, quality of service, deep packet inspection, and wire-speed switching and routing. His advisor is Professor Ying-Dar Lin, and his mentor is Dr. Cheng-Yuan Ho.

The possible upper bound of DCR is the DCR of Skype and there is no space to enhance NTTs if we cannot change the NAT infrastructures. On the other hand, we suggest more NAT infrastructure using independent mapping rules.