Behavior and Classification of NAT Devices and Implications for NAT Traversal

Andreas Müller and Georg Carle, Technische Universität München
Andreas Klenk, Universität Tübingen

Abstract

For a long time, traditional client-server communication was the predominant communication paradigm of the Internet. Network address translation devices emerged to help with the limited availability of IP addresses and were designed with the hypothesis of asymmetric connection establishment in mind. But with the growing success of peer-to-peer applications, this assumption is no longer true. Consequently network address translation traversal became a field of intensive research and standardization for enabling efficient operation of new services. This article provides a comprehensive overview of NAT and introduces established NAT traversal techniques. A new categorization of applications into four NAT traversal service categories helps to determine applicable techniques for NAT traversal. The interactive connectivity establishment framework is categorized, and a new framework is introduced that addresses scenarios that are not supported by ICE. Current results from a field test on NAT behavior and the success ratio of NAT traversal techniques support the feasibility of this classification.

When the Internet Protocol (IP) was designed, the growth of the Internet to its current size was not imaginable. Therefore, it was reasonable to use a fixed 32-bit field to identify a host based on its IP address. This limited address range makes it impossible to assign globally unique IPv4 addresses to the growing number of networked devices. Furthermore, requesting an IP address for every newly added device results in an unacceptable administration overhead. The authors in [1] propose to assign a number of public IP addresses to a designated border router instead of configuring certain hosts with addresses that can be routed globally. The border router is then responsible for translating IP addresses between the private and the public domains, allowing as many simultaneous connections as public IP addresses were assigned. This allows a host within the local network to access the Internet even though it has a private IP address. This technique became known as network address translation (NAT). Because the translation of addresses breaks the end-to-end connectivity model of the IP, newly developed services following the peer-to-peer (P2P) paradigm such as file sharing, instant messaging, and voice over IP (VoIP) applications suffer from the existence of NAT. Thus, NAT traversal is an important problem today. And even in the future, after a possible success of IPv6, companies and home users still might deploy NAT devices to hide their topologies from Internet service providers (ISPs). There are two possible approaches to the problem. One direction within the Internet Engineering Task Force (IETF) Behave Working Group [2] is to cope with existing NAT implementations and to establish standards for the detection of NAT behavior and for NAT traversal. On the other hand, the IETF also standardizes behavioral properties for NATs to work in conjunction with IETF protocols (e.g., Datagram Congestion Control Protocol [DCCP], Internet Control Message Protocol [ICMP], Stream Control Transmission Protocol [SCTP]). Enterprise class NATs are among the first to incorporate new features introduced through standardization. However, the large scale deployment of residential gateways with NAT functionality prohibits the change of NAT and requires the use of protocols that work with existing NATs. This is also the focus of this article, where we treat NATs as black boxes rather than trying to change them.

NAT Behavior

Today, a NAT device usually is used to share a single public IP address among a number of private end systems. The NAT maintains a table, listing all connections between the public and the private domains. For every connection attempt (e.g., a Transmission Control Protocol synchronize [TCP SYN] packet) coming from an internal host, the NAT creates a new entry in the list. In NAT terminology this entry is called a binding [3]. Each entry contains the source IP address and the source port. The NAT replaces the source IP address with its public IP address. The source port is replaced using one of the strategies explained later in this section.

Although the concept of NAT was published as early as 1994 [1], no common approach for NAT emerged. Current NAT implementations not only differ from vendor to vendor but also from model to model, which leads to compatibility

0890-8044/08/$25.00 © 2008 IEEE

Classification	NAT property
Port binding	Port preservation No port preservation Port overloading Port multiplexing
NAT binding	Endpoint-independent Address- (port)-dependent Connection-dependent
Endpoint filtering	Independent Address restricted Address and port restricted

Table 1. *NAT behavior categories and possible NAT properties.*

issues. If an application works with one particular NAT, this does not imply that it always works in a NATed environment. Therefore, it is very important to understand and classify existing NAT implementations in order to design applications that can work in combination with current NATs. The classification in this article is mainly derived from simple traversal of User Datagram Protocol (UDP) through NAT (STUN) [4], whereas the address binding and mapping behavior follows the terminology used in RFC 4787 [5]. This section covers only topics that are required for the understanding of this article. A detailed discussion and further information (including test results) is given in [6] (for TCP) and [5] (for UDP).

Binding covers "context based packet translation" [7], which describes the strategy the NAT uses to assign a public transport address (combination of IP address and port) to a new state in the NAT. Filtering, or packet discard, shows how the NAT handles (or discards) packets trying to use an existing mapping. Table 1 shows the different categories and their possible properties. Port binding describes the strategy a NAT uses for the assignment. With *port preservation*, the NAT assigns an external port to a new connection; it attempts to preserve the local port number if possible. Port overloading is problematic and rarely occurs. A new connection takes over the binding, and the old connection is dropped. Port multiplexing is a very common strategy where ports are demultiplexed based on the destination transport address. Incoming packets can now carry the same destination port and are distinguished by the source transport address.

NAT binding deals with the reuse of existing bindings. That is, if an internal host closes a connection and establishes a new one from the same source port, NAT binding describes the assignment strategy for the new connection. As shown in Table 1, the NAT binding is organized into three categories. With *Endpoint Independent*, the external port is only dependent on the source transport address of the connection. As long as a host establishes a connection from the same source IP address and port, the mapping does not change. The assignment is dependent on the internal and the external transport address with the *Address (Port) Dependent* strategy. As long as consecutive connections from the same source to the same destination are established, the mapping does not change. As soon as we use a different destination, the NAT changes the external port. With a *Connection Dependent* binding, the NAT assigns a new port to every connection. We distinguish between NATs that increase the new port number by a specific (and well predictable) delta and NATs that assign random port numbers to the new mappings.

Endpoint filtering describes how existing mappings can be used by external hosts and how a NAT handles incoming connection attempts that are not part of a response. *Independent Filtering* allows inbound connections independent of the source transport address of the packet. As long as the destination transport address of a packet matches an existing state, the packet is forwarded. With *Address Restricted Filtering*, the NAT forwards only packets coming from the same host (matching IP address) to which the initial packet was sent. *Address and Port Restricted Filtering* also compares the source port of the inbound packet in addition to address restricted filtering.

NAT Traversal Problem

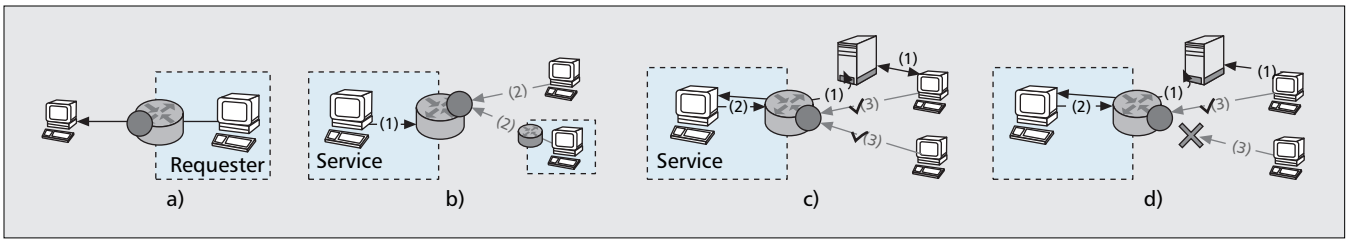
To work properly, the NAT must have access to the protocol headers at layers 3 and 4 (in case of a network address port translation [NAPT]). Additionally, for every incoming packet, the NAT must already have a state listed in its table. Otherwise, it cannot find the related internal host to which the packet belongs. According to RFC 3027 [8], the NAT traversal problem can be separated into three categories, which are presented in this section. In addition to the three problems, we identified *Unsupported Protocols* as a new category.

The first problem occurs if a protocol uses *Realm-Specific IP Addresses* in its payload. That is, if an application layer protocol such as the Session Initiation Protocol (SIP) uses a transport address from the private realm within its payload signalizing where it expects a response. Because regular NATs do not operate above layer 4, application layer protocols typically fail in such scenarios. A possible solution is the use of an application layer gateway (ALG) that extends the functionality of a NAT for specific protocols. However, an ALG supports only the application layer protocols that are specifically implemented and may fail when encryption is used.

The second category is *P2P Applications*. The traditional Internet consists of servers located in the public realm and clients that actively establish connections to these servers. This structure is well suited for NATs because for every connection attempt (e.g., a TCP SYN) coming from an internal client, the NAT can add a mapping to its table. But unlike client-server applications, a P2P connection can be initiated by any of the peers regardless of their location. However, if a peer in the private realm tries to act as a traditional server (e.g., listening for a connection on a socket), the NAT is unaware of incoming connections and drops all packets. A solution could be that the peer located in the private domain always establishes the connection. But what if two peers, both behind a NAT, want to establish a connection to each other? Even if the security policy would allow the connection, it cannot be established.

The third category is a combination of the first two. *Bundled Session Applications*, such as File Transfer Protocol (FTP) or SIP/Session Description Protocol (SDP), carry realm-specific IP addresses in their payload to establish an additional session. The first session is usually referred to as the control session, whereas the newly created session is called the data session. The problem here is not only the realm-specific IP addresses, but the fact that the data session often is established from the public Internet toward the private host, a direction the NAT does not permit (e.g., active FTP).

Unsupported Protocols are typically newly developed transport protocols such as the SCTP or the DCCP that cause problems with NATs even if an internal host initiates the connection establishment. This is because current NATs do not have built-in support for these protocols. The unsupported protocols also cover protocols that cannot work with NATs because their layer 3 or layer 4 header is not available for translation. This happens when using encryption protocols such as IPSec.



■ Figure 1. *NAT traversal service categories for applications: a) RNT; b) GSP; c) SPPS; d) SSP.*

NAT Traversal Service Categories

Instead of classifying the NAT behavior (see classification in STUN [4]), we defined four NAT traversal service categories, each making different assumptions about the purpose of the connection establishment and the infrastructure that is available. Our categorization emphasizes that the applicability of many NAT traversal techniques depends on the support of a combination of requester, the responder, globally reachable infrastructure nodes, and the role of the application. On the one hand, server applications set up a socket and wait for connections (which also applies to P2P applications). On the other hand, client applications such as VoIP clients actively initiate a connection and wait for an answer on a different port (bundled session applications). Other applications work only across NATs if both ends participate in the connection establishment (unsupported protocols). Thus, we differentiate between supporting a service and supporting a client. In this article, the client is called the requester because it actively initiates a connection.

The behavior of the NAT is important because it allows or prohibits certain NAT traversal techniques within one service category. If only one end implements NAT traversal support (e.g., by running a stand-alone framework or by built-in NAT traversal functionality), NAT traversal techniques that rely on a collaboration of both ends (e.g., ICE) are not applicable.

Our first category, requester side NAT traversal (RNT), covers scenarios where only the requester side supports NAT traversal (e.g., the application or the NAT itself). RNT helps applications that actively participate in the connection establishment and still suffer from the existence of NATs. Typical examples are applications that have problems with realm-specific IP addresses in their payload. This applies to protocols using in-band signaling on the application layer, which is related to bundled session applications with asymmetric connection establishment (e.g., VoIP using SIP/SDP).

The second category, global service provisioning (GSP), assumes that the host providing the service implements NAT traversal support, helping to make a service globally accessible. This is done by creating and maintaining a NAT mapping that then accepts multiple connections from previously unknown clients (Fig. 1). This is the main difference from RNT, which only creates a NAT mapping for one particular session (e.g., one call in the case of VoIP).

The last two categories assume support at both ends, the service and the requester. On the one side, NAT traversal is required to make a service behind a NAT globally accessible, whereas on the other side, the support at the requester allows the use of sophisticated techniques through coordinated action. Thus, service provisioning using pre-signaling (SPPS) extends the GSP category by the assumption that both hosts have interoperable frameworks (e.g., ICE [9]; NAT, URIs, Tunnels, SIP, and STUNT [NUTSS] [10]; NATBlaster [11]; or NatTrav [12]) running. This allows a selection from all available NAT traversal solutions, which leads to a high success rate of NAT traversal. In Fig. 1, the two hosts use a rendezvous point to agree on a NAT traversal technique. After creating the mapping in step 2, the service is accessible by any host, depending on the selected NAT traversal technique and the filtering strategy of the NAT. SPPS supports all types of services where a one-to-one connection is sufficient and pre-signaling is available.

The last category, secure service provisioning (SSP), is an extension of SPPS and addresses scenarios that require authorization of the remote party before initiating the NAT traversal process. The hereby established channel must be accessible only by the authorized remote party. This requires additional functionality that enforces this policy and only allows authorized users to access the service. The policy enforcement can be done at the NAT itself, at a data relay, or at a firewall. Table II depicts all four service categories with popular NAT traversal techniques and shows the implications for automated NAT traversal and required signaling. First we distinguish between the service and the requester. "Support at the service" means, for example, that a framework must be deployed at the same host providing the service. The same applies to the requester. "RP" means that a rendezvous point is required for relaying data back and forth. "Signaling messages" means that some sort of signaling protocol is used for NAT traversal. Again, we differentiate between signaling at the service and signaling at the requester. A rendezvous point for signaling messages is required in case of pre-signaling. Finally, "stream independent" describes the requirement for consecutive connections. For example, a port forwarding entry must be created only once, whereas hole punching [13] requires sending a new hole punching packet for every new stream (with restricted filtering).

Table 2 shows the main differences of our service categories. RNT deals with bundled session applications that wait on a port after initiating a session (e.g., via a SIP INVITE). GSP requires only support of the service and aims to make a service globally reachable for multiple clients. SPPS and SSP combine these categories and require support at both ends. The requester initiates pre-signaling to exchange information about a global end point. The service then creates a mapping in the NAT that can be used by the client.

Applicability of NAT Traversal Techniques for NAT Traversal Service Categories

There are many different techniques for solving the NAT traversal problem in specific scenarios, but none of them provides a solution that works well with all NATs, applications, and network topologies. Another article explains many of the available protocols for NAT traversal [14] in general. This section describes the applicability of existing techniques from the applications point of view.

RNT is required for protocols using in-band signaling (bundled session applications). Therefore, one common approach is to integrate RNT into these applications (e.g., the VoIP client), to establish port bindings on the fly. One possibility is the integration of a universal plug and play (UPnP) client. Another option is to use ALGs that are integrated in the NAT, interpreting in-band signaling and establishing map-

Service category	NAT traversal techniques	Requires support at				Signaling messages				Stream-independent
		Service	Requester	RP	NAT	Service	Requester	RP	STUN	
RNT	NAT with ALG				X					
	UPnP (for bundled session applications)		X		X		X			X
GSP	UPnP (port forwarding)	X			X	X				X
	Hole punching — independent filtering	X				X			X	X
	Open data relay (e.g., RSIP)	X		X		X		X		X
SPPS	Hole punching — independent binding	X	X			X	X	X	X	
	UPnP	X	X		X		X	X	X	
	Closed/open data relay (e.g., TURN, Skype)	X	X	X		X	X	X		
	Tunneling (e.g., over UDP)	X	X			X	X			X
SSP	Hole punching — restricted filtering	X	X			X	X	X	X	
	NSIS NATFW NSLP	X	X		X	X	X			
	Closed data relay (e.g., TURN)	X	X	X		X	X	X		
	Tunneling (e.g., over secure channel)	X	X			X	X			X

■ Table 2. *Service categories and their implications for automated NAT traversal; RP denotes rendezvous point.*

pings accordingly. ALGs are not a general solution because the NAT must implement the required logic for each protocol, and end-to-end security prohibits the interpretation of the signaling by the NAT.

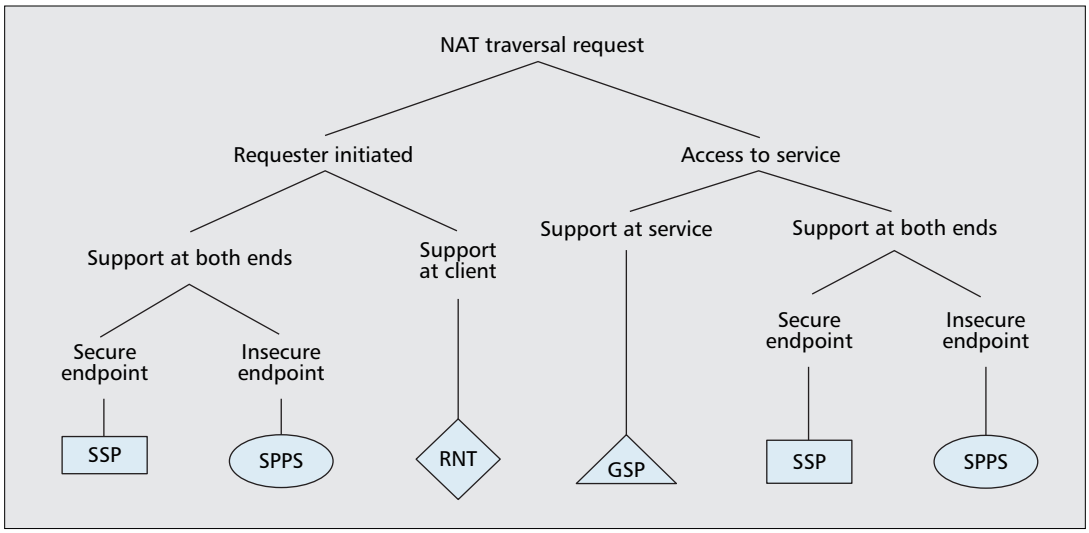
GSP depends on NAT traversal techniques that allow unrestricted access to a public end point. A control protocol can be used to directly establish a port forwarding entry in the mapping tables of the NAT, for instance, with UPnP [15]. Port forwarding entries created by UPnP are easy to maintain and work independently from NAT behavior. However, UPnP only works if the NAT is in the local network on the path to the other end point. Thus, nested NATs are not allowed, and path changes break the connectivity.

Hole punching is an alternative if UPnP is not applicable and works for NATs with an independent filtering strategy. The mapping must be refreshed periodically, for instance, by sending keep-alive packets. For NATs other than full-cone, hole-punching for GSP cannot be used because the source port of the request is unknown in advance.

~~SSPS~~ makes no assumption about the accessibility of a created mapping, thus all possible techniques are applicable. Different from GSP, hole-punching for SPPS works as long as port prediction is possible. For NATs implementing restricted filtering, pre-signaling helps to create the appropriate mapping because the five-tuple of the connection is exchanged. Pre-signaling also enables the establishment of an UDP tunnel, allowing the encapsulation of unsupported protocols. SPPS also can use UPnP to establish port forwarding entries for one session.

SSP is an extension to SPPS that allows only authorized hosts to allocate and to use a mapping. Protocols that authorize requests and assume control over the middlebox, such as middlebox communication (MIDCOM) [16] or the NAT/Firewall Next Step in Signaling (NSIS) Layer Protocol [17] qualify for SSP. The advantage of NSIS is that it can discover and configure multiple middleboxes along the data path, thus supporting complex scenarios with nested NATs and multipath routing. However, if one NAT on the path does not support the protocol, NSIS fails. Using NSIS and MIDCOM for SSP requires restrictive rules that allow only authorized clients to use the mapping, for instance, by opening pinholes for IP five-tuples. UPnP is not useful for SSP because it forwards inbound packets without considering the source transport address. Hole punching can be used only with SSP if the NAT implements a restricted filtering strategy. All cases discussed previously rely on additional measures to prohibit IP spoofing. The use of secure tunnels impedes IP spoofing and allows secure NAT traversal, even for unsupported protocols (e.g., IPSec, SCTP, DCCP). SSP also can be achieved by using traversal using relay NAT (TURN) with authentication, authorization, and secure communication (e.g., via transport layer security [TLS]).

ICE [9] is under standardization by the IETF and strives to combine several techniques into a framework flexible enough to work with all network topologies. Because ICE requires both peers to have an ICE implementation running, it can be seen as a technique for SPPS or SSP, depending on the accessibility and the security policies of the public endpoint.



■ Figure 2. *Decision tree for ANTS.*

The same is true for solutions such as TURN [18]. TURN is a promising candidate for SPPS, because it provides a relay with a public transport address allowing the exchange of data packets between a TURN client and a public host.

Why Unilateral Solutions Exist

Despite the great flexibility of SPPS and SSP, both categories involve a number of assumptions that are not always satisfied. The most important one is the requirement for both ends (and sometimes also the infrastructure), to support compatible versions of the NAT traversal framework. It remains to be seen if the future will bring a sufficiently big deployment of one framework on which to rely for arbitrary applications. The chances are better within homogeneous problem domains, like telecommunication, where such frameworks can be integrated with the applications and be distributed in large numbers. For instance, the adoption of ICE is occurring mainly within the VoIP/SIP community and focusing on VoIP specific use cases. These drawbacks are the reason why RNT and GSP as unilateral solutions for the NAT traversal problems exist. It is easier to enhance an infrastructure under one responsibility than to rely on a solution that requires a global deployment. However, unilateral solutions are limited to the middleboxes in the given domain. They fail to provide solutions to scenarios with nested NATs and depend on the network topology.

Coalescing Unilateral and Cooperative Approaches for NAT Traversal

When investigating existing NAT traversal techniques, we determined that none of them can be used in all scenarios. For example, UPnP only supports globally accessible end points, whereas ICE requires both hosts to run the framework. In [19], we proposed a new framework that aims toward providing an advanced NAT traversal service (ANTS) supporting all four service categories. The concept of ANTS is based on the idea of reusing previously obtained knowledge about the topology of the network and the capability of the NAT. A small component of ANTS, the NAT tester, is responsible for gathering this information and will be presented (together with some test results) in the next section.

If a user decides that a particular application should be reachable from the public Internet, he registers it at a session manager that keeps track of all applications requesting NAT traversal support. With the session manager, ANTS can provide GSP and RNT directly. Whenever an application is added and associated with GSP or RNT, the session manager calls the NAT traversal logic and asks to allocate an appropriate mapping in the NAT. This also requires ANTS to have sufficient knowledge about the applicability of the integrated techniques regarding the service categories. For example, UPnP cannot be used for SSP because it violates the idea of an endpoint that is accessible only by authenticated hosts.

Figure 2 shows a decision tree that ANTS uses to establish a mapping in the NAT. First, we distinguish between requester initiated NAT traversal on the one hand and the access to a service on the other hand. Then, we must know which ends actually implement ANTS. If both hosts have the framework running, pre-signaling is possible, which leads to a wide choice of techniques depending on the security considerations of the mapping. If only one end supports ANTS, only techniques belonging to GSP or RNT are applicable.

Despite some unsolved issues such as the question of how to connect legacy applications to ANTS (e.g., by using a library or a traversal of UDP through NAT [TUN]-based approach), the idea of a knowledge-based framework seems

S. cat.	Prot.	Condition	Suc. rate
RNT	UDP	(UPnP *or* HP-UDP)	90.27%
	TCP	(UPnP *or* HP-TCP)	77.84%
GSP	UDP	(Full Cone and HP-UDP)	27.03%
	TCP	(Full Cone and HP-TCP)	17.30%
	UDP	(UPnP or (Full Cone and HP-UDP))	50.27%
	TCP	(UPnP or (Full Cone and HP-TCP))	44.32%
SPPS	UDP	(HP-UDP)	88.65%
	TCP	(HP-TCP)	71.35%
	TCP	(HP-TCP or HP-UDP)	94.59%
	UDP	(UPnP or HP-UDP)	90.27%
	TCP	(UPnP or HP-TCP)	77.84%
	TCP	(UPnP or HP-TCP or HP-UDP)	95.14%
SSP	UDP	(Restricted NAT and HP-UDP)	48.65%
	TCP	(Restricted NAT and HP-TCP)	38.38%

■ Table 3. *Results of the field test: success rates of NAT traversal techniques depending on service categories.*

to be the right answer. Thus once implemented, ANTS can help many existing services by integrating several techniques and making its choice based on knowledge about the NAT and the requirements of the application.

Field Test on NAT Traversal

To prove that existing techniques can be adapted to our service categories, we implemented a NAT tester that acts as a cornerstone for our new framework. This section presents the results of a field test investigating 185 NATs in the wild. For a detailed description including all results, see our Web site: http://nettest.net.in.tum.de.

The first test queries a public STUN server to determine the type of the NAT. Afterward, the NAT tester performs the following connection tests and tries to establish a connection to the host behind the NAT: UPnP, hole punching, and connecting to a data relay (each for both protocols, UDP and TCP) (Table 3).

We then adapted the test results to our work and evaluated the success rates of the individual techniques regarding our defined service categories. Table III shows the categories and the conditions that must be met according to the considerations made previously. For example, GSP requires the use of UPnP or hole punching support in combination with a full-cone NAT to make a service globally accessible. Therefore, 50.27 percent of our tested NATs supported a direct connection for UDP and category GSP (44.32 percent for TCP). In all other cases (the remaining percentages), an external relay must be used to provide GSP.

For SPPS, which makes no security assumptions, we divided our results into two categories. First we determined the success rates without considering UPnP. With 88.65 percent of all NATs, we were able to establish a direct connection to the host behind the NAT (71.35 percent for TCP). This rate increased slightly (for TCP to 77.84 percent) when UPnP was an option. The highest success rate for TCP NAT traversal (95.14 percent) was discovered when we also allowed the tunneling of TCP packets through UDP.

SSP allows only authorized hosts to create and to use a mapping. Therefore, a suitable technique for SSP is hole punching in combination with a NAT implementing a restricted filtering strategy. This was supported by 48.65 percent for UDP and 38.38 percent for TCP.

The success rate for RNT depends on the effort that is made for the specific protocol. For example, if we assume that we can inspect each signaling packet on the application layer thoroughly, we could adopt the results from SPPS to RNT. If we would only modify the packets in a way that the internal port is reachable by any client, the success rate of GSP would apply to RNT. Finally, we did not measure the effect of NATs with integrated ALGs in this field test.

Conclusion

With the increasing popularity of P2P communication, the NAT traversal problem has become more urgent than ever. Existing solutions have the drawback of supporting only certain types of NATs and cannot be viewed as a general solution to the problem. When analyzing the NAT traversal problem more thoroughly, we discovered that the question of who supports the NAT traversal framework determines which NAT traversal techniques are applicable. Therefore, we identified four NAT traversal service categories that differentiate between support by service, client, and infrastructure and listed applicable NAT traversal techniques for each category. Our findings from a field test showed that there are a number of prospective NAT traversal techniques that enable connectivity for each NAT traversal service category. We emphasized how to build upon this categorization to develop a knowledge-based NAT traversal framework. Future frameworks that aspire to support the typical connectivity scenarios of current applications should support all four service categories.

References

[1] K. Egevang and P. Francis, "The IP Network Address Translator (NAT)," IETF RFC 1631, May 1994.
[2] IETF, "Behavior Engineering for Hindrance Avoidance (behave);" http://www.ietf.org
[3] P. Srisuresh and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations," IETF RFC 2663, Aug. 1999.
[4] J. Rosenberg *et al.*, "STUN: Simple Traversal of User Datagram Protocol (UDP) through Network Address Translators (NATs)," IETF RFC 3489, Mar. 2003.
[5] E. F. Audet and C. Jennings, "NAT Behavioral Requirements for Unicast UDP," IETF RFC 4787, Jan. 2007.
[6] S. Guha and P. Francis, "Characterization and Measurement of TCP Traversal through NATs and Firewalls," *Proc. ACM Internet Measurement Conf.*, Berkeley, CA, Oct. 2005.
[7] G. Huston, "Anatomy: A Look Inside Network Address Translators," *The Internet Protocol J.*, vol. 7, 2004, pp. 2–32.
[8] M. Holdrege and P. Srisuresh, "Protocol Complications with the IP Network Address Translator," IETF RFC 3027, Jan. 2001.
[9] J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols," IETF Internet draft, work in progress, Oct. 2007.
[10] P. Francis, S. Guha, and Y. Takeda, "NUTSS: A SIP-based Approach to UDP and TCP Network Connectivity," Cornell Univ., Panasonic Commun., tech. rep., 2004.
[11] A. Biggadike *et al.*, "NATBLASTER: Establishing TCP Connections between Hosts behind NATs," *ACM SIGCOMM Asia Wksp.*, Beijing, China, 2005.
[12] J. Eppinger, "TCP Connections for P2P Applications — A Software Approach to Solving the NAT Problem," Carnegie Mellon Univ., Pittsburgh, PA, tech. rep., 2005.
[13] B. Ford, P. Srisuresh, and D. Kegel, "Peer-to-Peer Communication across Network Address Translation," MIT, tech. rep., 2005.
[14] H. Khlifi, J. Gregoire, and J. Phillips, "VoIP and NAT/Firewalls: Issues, Traversal Techniques, and a Real-World Solution," *IEEE Commun. Mag.*, July 2006.
[15] U. Forum, "Internet Gateway Device (IGD) Standardized Device Control Protocol," Nov. 2001.
[16] P. Srisuresh *et al.*, "Middlebox Communication Architecture and Framework," IETF RFC 3303, Aug. 2002.
[17] M. Stiemerling *et al.*, "NAT/Firewall NSIS Signaling Layer Protocol (NSLP)," IETF Internet draft, Feb. 2008.
[18] J. Rosenberg, R. Mahy, and P. Matthews, "Traversal Using Relays around NAT (TURN)," IETF Internet draft, work in progress, June 2008.
[19] A. Müller, A. Klenk, and G. Carle, "On the Applicability of Knowledge-Based NAT-Traversal for Future Home Networks," *Proc. IFIP Networking 2008*, Springer, Singapore, May 2008.

Biographies

ANDREAS MÜLLER (mueller@net.in.tum.de) received his diploma degree in computer science from the University of Tübingen, Germany in 2007. Currently, he is a research assistant and Ph.D. candidate at the Network Architecture and Services Department at the Technical University of Munich. His research interests include middleboxes, P2P systems, and autonomic networking.

ANDREAS KLENK (klenk@informatik.uni-tuebingen.de) earned his diploma degree in computer science from Ulm University, Germany, in 2003. He is a Ph.D. candidate and research assistant at the University of Tübingen and works with Professor Carle. He contributes to European research projects in the telecommunication field. His research interests include negotiation and security in autonomic systems.

GEORG CARLE (carle@net.in.tum.de) received a M.Sc. degree from Brunel University London in 1989, a diploma degree in electrical engineering from the University of Stuttgart in 1992, and a doctoral degree from the faculty of computer science, University of Karlsruhe in 1996. He is a full professor in computer science at the Technical University of Munich, where he is chair of the Department of Network Architecture and Services. Among the focal interests of his research are Internet technology and mobile communication in combination with security.