On the Applicability of knowledge based NAT-Traversal for Home Networks

Andreas Müller, Andreas Klenk, and Georg Carle

University of Tübingen, Computer Networks and Internet,
Sand 13, 72076 Tübingen, Germany
`{mueller,klenk,carle}@ri.uni-tuebingen.de`
`http://net.informatik.uni-tuebingen.de`

Abstract. The presence of Network Address Translation (NAT) is a hindrance when accessing services within home networks, because NAT breaks the end-to-end connectivity model of the Internet protocol suite. Communication across NATs is only possible if it is initiated from a host belonging to the internal network. Thus, services expecting a connection established from the outside fail in most situations. Existing approaches for NAT-Traversal do not cover the full range of NAT-Traversal methods and fail in certain situations, or deliver sub optimal results in others. Part of the problem of existing approaches is that they do not differentiate between different types of applications. We argue that the classification of applications into four service categories helps to determine the best matching NAT-Traversal technique. An extensive field test enables us to acquire knowledge about the success rates of promising NAT-Traversal techniques. These results will help us to develop a knowledge driven NAT-Traversal framework making its choice based on an understanding of NAT behavior, NAT-Traversal options and the service category of the application.

Key words: NAT-Traversal, Field Test on NAT Behavior

1 Introduction

Today most home networks connect to the public Internet via a "Designated Border Router", allowing multiple clients to share one public IP address using Network Address Translation (NAT). We use the term NAT as defined in [1]: "a method by which IP addresses are mapped from one realm to another, in an attempt to provide transparent routing to hosts". Since NAT breaks the end-to-end connectivity model of the TCP/IP-protocol suite, a communication across a middlebox performing NAT is only possible if it is initiated from a host belonging to the internal network. Therefore, applications behind a Network Address Translator aiming to provide a globally reachable service (e.g. Web Servers, Peer-to-Peer or VoIP applications) suffer from the existence of NATs. This problem is known as the NAT-Traversal problem.

The NAT-Traversal problem arises as soon as an external host wants to connect to a peer located in the private network. This is because NAT only passes inbound packets to an internal host if it has a state listed in its mapping table. The creation of a state however depends on a packet traveling in the other direction than the incoming packet. In other words, before an internal host is able to receive packets, it first has to send a packet to the remote peer itself. NAT then tries to handle all incoming packets as a response to the outgoing packet and passes them to the appropriate internal host.

There are many different approaches for solving the NAT-Traversal problem, but none can claim to solve the problem in all situations without drawbacks. Some are behavior based and only support UDP (e.g. ICE [2]), while others introduce a significant communication overhead and require the presence of infrastructure nodes (e.g. TURN [3]). An alternative to behavior based approaches is to exercise direct control over the NAT, for instance by using UPnP [4] or MIDCOM [5] in order to establish port forwarding entries. There is a lack of extensible frameworks that can make a choice when to utilize behavior based or control based techniques.

Our aim is to establish persistent knowledge about the available NAT-Traversal options and to make an intelligent choice depending on the application. This paper has four main contributions: 1.) A survey on the behavior of Network Address Translators and techniques for NAT-Traversal 2.) Service categories for applications requiring NAT-Traversal support 3.) Framework proposal for knowledge based NAT-Traversal 4.) A Field test on the success rate and applicability of NAT-Traversal techniques regarding our four service categories.

The remainder of this paper is structured as follows. In Sec. 2 we rehash NAT behavior. Sec. 3 explains NAT-Traversal techniques. Service categories for applications are introduced in Sec. 4. Sec. 5 introduces the idea of the ANTS framework for NAT-Traversal. In Sec. 6 we present our field test on NAT behavior. Sec. 7 surveys related work. Finally, Sec. 8 summarizes the contributions of this paper.

2 NAT Behavior

Current NAT implementations not only differ from vendor to vendor, but also from model to model. If an application works with one particular Network Address Translator, there is no guarantee that it always works in a NATed environment. This section gives a short introduction to a common classification proposed by [6]. For a more detailed discussion please refer to [7] and [8]. A *Full-Cone NAT* uses an endpoint independent binding strategy (the same external endpoint is assigned to two consecutive connections from the same source transport address (the combination of IP address and port) independent from the destination transport address) and an independent filtering strategy. Thus, once a mapping is created, every external host can access it in order to communicate with the appropriate internal client. An *Address-Restricted-Cone NAT* uses an endpoint independent binding in combination with an address restricted filtering strategy: only packets that originate from the same host the initial packet has been sent to are forwarded. Again, endpoint independent binding is used for a *Port-Address-Restricted-Cone NAT*. In addition to the destination address, a Port-Address-Restricted-Cone NAT also considers the port of the external host when translating packets between the realms. Other than the three categories described above, a *Symmetric NAT* uses a non-independent mapping. Depending on the exact strategy (random or not), port prediction may not be possible. Symmetric NATs usually behave like Port-Address-Restricted-Cone NATs with respect to filtering.

3 Techniques for NAT-Traversal

This section presents popular and promising methods for NAT-Traversal. One important group of NAT-Traversal techniques assumes direct control over the NAT. Adding a *Port-Forwarding* entry to the Network Address Translator can be done manually, which is only feasible for experienced users, or dynamically. Controlling the Network Address Translator transparent to the user is the aim of protocols such as *Universal Plug and Play* (UPnP) [4], MIDCOM [5], NAT/Firewall NSIS Signaling Layer Protocol [9] or the NAT-Port-Mapping Protocol [10]. However, if the protocol is not explicitly supported by the NAT (e.g. UPnP may be disabled due to security issues) it fails. Furthermore, every external port can only be used once. If, e.g., two hosts in a private network want to provide a service running on the same port, only one can use port preservation.

There are also NAT-Traversal methods that do not require support by the Network Address Translator. These techniques try to anticipate NAT behavior in order to establish a connection. One such technique is UDP Hole-Punching. Hole-Punching was first described in [11] and more thoroughly documented in [12]. In order to receive packets behind a Network Address Translator, the initial packet coming from the remote host has to look like a response to the Hole-Punching packet sent by the NATed host. Simple Traversal of User Datagram Protocol through NATs (STUN) [6] allows to determinate external transport addresses and works well for endpoint independent mapping. Today, UDP Hole-Punching is used by many proprietary protocols for Instant-Messaging, Online-Gaming and VoIP applications.

Due to the connection-oriented design of TCP, Hole-Punching for TCP is more difficult than for UDP. The TCP NAT-Traversal method STUNT [8] requires a NAT to accept an incoming TCP-SYN packet following an outgoing TCP-SYN packet (the outgoing TCP-SYN being the Hole-Punching packet), a sequence of packets usually not seen and therefore often discarded.

Hole-Punching is not a suitable technique for NAT-Traversal of Symmetric NATs, since port prediction usually fails. The goal of Traversal Using Relays around NAT (TURN) [3] is to provide a relay with a public transport address allowing the exchange of data packets between a TURN-client and a public host. TURN has the advantage of working with almost every NAT because all incoming packets are directly related to outgoing packets. As a drawback, the reliability and the bandwidth of a connection depends on the chosen TURN server, introducing a Single Point of Failure.

4 Service Categories for NAT-Traversal

When investigating existing applications suffering from the presence of NAT, we identified four service categories an application can belong to. Each category addresses different requirements and makes assumptions about the network topology and available components.

Our first category, **Global Service Provisioning (GSP)**, assumes that only one host has a NAT-Traversal framework running (no signaling is needed). It helps to make a local service globally accessible by creating and maintaining a NAT-mapping (e.g. sending keep-alive packets). Services such as a Web-Server belong to GSP. Peers of P2P content distribution networks can establish direct connections, even if both peers are behind a NAT and one of them supports GSP.

Service Provisioning using Pre-Signaling (SPPS) extends the GSP-category by giving NAT-Traversal support to both parties, which allows more sophisticated connection establishment procedures such as signaling. Compared to related work, SPPS is the category used by most existing frameworks [2][13][14]. The advantage of SPPS is that the NAT-Traversal service can select from all available NAT-Traversal solutions. Therefore, SPPS should be used, if possible, because it provides the highest success rate regarding NAT-Traversal.

Our third category, **Secure Service Provisioning (SSP)**, is an extension to SPPS and addresses applications that only want to allow authorized hosts to create and use a mapping in the Network Address Translator. Many applications were originally designed to run in a closed environment only, such as a home or company network. Such services do not always realize a high level of security, making them a target for hackers if they are reachable from the public Internet. Unfortunately, none of the NAT-Traversal solutions available today actually offers restricted access to certain services.

The last category, **ALG Service Provisioning (ALG-SP)**, focuses on applications having problems with realm-specific IP-Addresses in their payload. This applies to protocols using inband-signaling on the application layer. This is also strongly related to Bundled-Session Applications with asymmetric connection establishment establishing separate control- and data-connections. The most popular application belonging to this category is VoIP using SIP. Besides ICE, SIP NAT-Traversal is usually solved by STUN or a SIP-aware Network Address Translator. But STUN only works with an independent mapping strategy, and SIP-aware NATs are not widely used. Therefore, in addition to the three categories described above, a NAT-Traversal service should also be able to handle protocols using inband-signaling without relying on a NAT-ALG.

5 Concept for a NAT-Traversal Framework

This section presents the idea of ANTS, a new framework which aims towards providing an Advanced NAT-Traversal Service for legacy applications. We first give an overview by describing a particular scenario for NAT-Traversal in section

5.1. Section 5.2 then discusses the applicability of existing NAT-Traversal techniques regarding our service categories and explains our main idea of a knowledge based framework.

5.1 Towards the Advanced NAT-Traversal Service

ANTS is deployed directly at the host and is aware of all registered applications in order to provide NAT-Traversal support based on the requested service category.

Figure 1 shows a scenario where two hosts, both behind NAT, want to establish a direct connection to each other. First, the ANTS-module gathers knowledge about its environment and determines which NAT-Traversal techniques can be used in order to traverse the Network Address Translator it is behind (see 5.2).

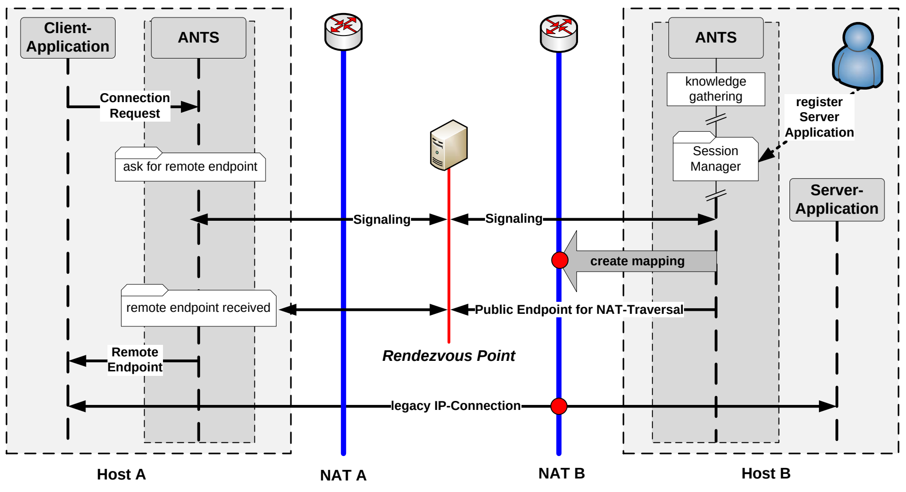

Fig. 1. Scenario for a new NAT-Traversal framework

If a user decides that a particular application should be reachable from the public Internet, he registers it at a Session Manager keeping track of all applications and their service categories. With the Session Manager, ANTS is able to provide Global Service Provisioning directly. Whenever an application is added and associated with GSP, the Session Manager calls the NAT-Traversal logic and asks to allocate an appropriate mapping in the NAT (for GSP, only the host that wants to offer services must run ANTS).

SPPS and SSP, however, require some form of signaling in order to exchange connection specific information in advance. Signaling should result in a working public transport address for NAT-Traversal. In order to avoid a Single Point of

Failure, signaling should be done in a decentralized way (e.g. Peer-to-Peer SIP (P2P-SIP) [15]). Furthermore, we also aim towards port preservation, meaning that the legacy applications only use realm-specific ports (e.g. port 80 for a web-server) independent of the global port allocated by the NATs.

5.2 Knowledge based NAT-Traversal

The concept of ANTS is based on the idea of reusing previously obtained knowledge. This means, instead of querying the network on every new connection request, ANTS determines the properties and capabilities of the NAT periodically, resulting in a very fast connection establishment.

The Knowledge and Decision Module not only needs to determine all working NAT-Traversal techniques, it also has to make sure that ANTS only applies NAT-Traversal techniques suitable for the requested service category. For example, UPnP cannot be used with Secure Service Provisioning because it violates the idea of a secure public endpoint. Therefore, every NAT-Traversal technique integrated into ANTS has to be associated with an entry as follows:

$$\textit{(Identifier} \rightarrow \textit{Service Category} \rightarrow \textit{Condition)}$$

The last field describes the conditions that have to be met for the technique in order to work with the defined service category. Global Service Provisioning depends on NAT-Traversal techniques that allow unrestricted access to a public endpoint. A Port-Forwarding entry created by UPnP is easy to maintain and works independently from NAT behavior. Hole-Punching for GSP is also possible if the NAT is of type Full-Cone. Finally, an independent relay can be used as the third option.

Since SPPS makes no assumptions about the accessibility of the mapping, there are no restrictions to be considered. As long as the technique is supported it may be used. As SPPS uses a signaling protocol between two hosts, more sophisticated techniques such as tunneling TCP within UDP are also possible.

SSP is an extension to SPPS and relies on the same signaling infrastructure. Since it only allows authorized hosts to allocate and to use a mapping, we have to choose from the following techniques: Hole-Punching with a restricted filtering strategy or TURN (which behaves like a Restricted Cone-NAT) are possible.

The applicability of NAT-Traversal techniques for ALG-SP depends on the security considerations made by the user for the particular service. As long as we only want to provide NAT-Traversal based on SPPS, any approach can be used. On the other hand, if we prefer to only allow certain hosts to access the NAT-mapping, the considerations made above when discussing SSP also apply to ALG-SP.

Figure 2 shows the components necessary for a knowledge based framework. When making a decision (e.g. which NAT-Traversal technique should be used for the requested application) the Knowledge and Decision Module relies on input from an overlaying Input Module. Besides the rather static knowledge about the applicability of existing NAT-Traversal techniques, the Knowledge and Decision Module should also consider user input, as well as input from the Session Manager as described above. However, the most important component of the Input Module has to be able to test the Network Address Translator regarding its core properties and the supported NAT-Traversal methods. Only if ANTS is able to determine the properties of the NAT correctly, it is able to reuse its knowledge when providing a working public endpoint to an application.

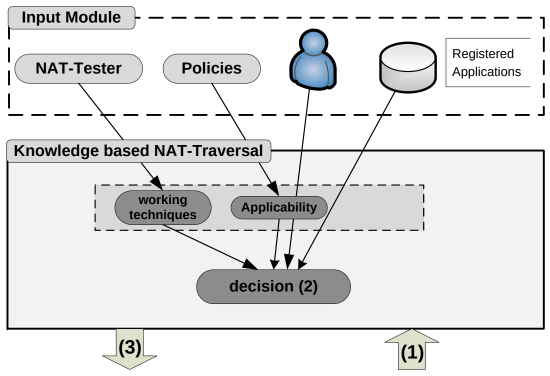

Fig. 2. Inputs to be considered in order to make decisions

6 Knowledge Gathering for NAT-Traversal

We described the idea of ANTS, a knowledge based approach for solving the NAT-Traversal problem in section 5. The fundamental question is how to discover the behavior of individual NAT-Traversal techniques in order to establish this knowledge. Throughout the whole paper we stated that the implementation of NAT is not standardized and differs from model to model. To get an overview of the behavior of current NAT implementations we devised the NAT-Tester as a module that can work as a stand-alone program and as an integral part of ANTS at the same time. We then did a field test investigating the behavior of 104 NATs in the wild, with our NAT-Tester available here: *http://gex.cs.uni-tuebingen.de*. Similar tests have been done in the past [8] [16], but all of them only tested a few NAT properties. For example, the combination of UPnP and different Hole-Punching techniques has never been considered. Additionally, the results are dependent on the market share of the NAT manufacturers, which differs from country to country. While all former results represent the North American Market, our test mainly grasps the realities in the German market.

6.1 NAT Behavior Discovery

As the first test, the NAT-Tester queried a public STUN-Server and determined the properties of the NAT (Table 1). More than *85%* of all NATs were either of type *Port-Address Restricted* or *Full-Cone*. Symmetric NATs, which are said to be popular in the business market, were rarely discovered. The second test then queried the network to determine if the NAT supports UPnP. According to Table 1, *51.92%* of all NATs had UPnP disabled. *No IGD* means that the NAT-Tester found an UPnP-capable device on the private network, but it was not an Internet-Gateway-Device (IGD).

Table 1. Results of the Field Test: NAT behavior

NAT Type	Appearance
Port-Address Restricted	50.96 %
Full-Cone	36.54 %
Address-Restricted	4.81 %
Symmetric	4.81 %
No-NAT	1.92 %
Symmetric Firewall	0.96 %

Behavior	Appearance
Port Preservation	63.46 %
Hairpinning	51.92 %
no UPnP	51.92 %
no IGD	9.62 %
UPnP ok	38.46 %

6.2 Anticipated Success of NAT-Traversal techniques

The next tests actually determined the success rates for different NAT-Traversal techniques. First, the NAT-Tester created a Candidate-List (listing all public endpoints to be tested) for UDP containing a STUN-derived public transport address for Hole-Punching. For the Hole-Punching test, the NAT-Tester sent a UDP packet towards the Test-Server allocating an appropriate mapping in the NAT. The NAT-Tester then started a listener on the private port and waited for a connection from the Test-Server. If a packet was received within a limited amount of time, the test succeeded. Table 2 shows the success rates for UDP. While *93.67%* of our 104 NATs supported UDP-Hole-Punching via STUN, the success rate of UPnP was only *38.46%*.

The UDP test algorithm was adapted to TCP and showed interesting results. In [8], the authors identified Hole-Punching using a Hole-Punching packet with a low TTL to be the most promising method with a success rate of *88%* and rely on it for their NAT-Traversal technique STUNT2. In our tests this form of TCP-Hole-Punching only succeeded for *51.92%* of all NATs. After discovering that the actual success rate was much lower, we modified the NAT-Tester and added two other types of TCP-Hole-Punching to it[1]. *HP High-TTL* uses a high TTL for the Hole-Punching packet resulting in a *RST*-packet from the

[1] 43 NATs were tested with the modified NAT-Tester

Table 2. NAT-Traversal for UDP

Technique	Appearance
Relay	100 %
Hole-Punching	92.31 %
UPnP	38.46 %

Table 3. NAT-Traversal for TCP

Technique	Appearance
Relay	100 %
HP FTP-ALG	67.44 %
HP Low-TTL	51.92 %
UPnP	38.46 %
HP High-TTL	13.95 %

NAT. With a success rate of only *13.95%*, it cannot be seen as a general solution for TCP-NAT-Traversal. *HP FTP-ALG* [17] relies on a working ALG for active FTP. The ALG creates mappings in the NAT according to the FTP signaling messages. But instead of providing a working FTP endpoint within the control messages, an independent transport address is listed. This unconventional technique works with *67.44%* of all NATs. For future tests it could be interesting to extend this idea to SIP and to provide a transport address within the SDP message.

Table 4. Results of the Field Test: Success rates for the different Service Categories

Category	Condition	Success Rate
GSP1 (UDP)	(Full-Cone *and* HP-UDP)	35.58 %
GSP3 (TCP)	(Full-Cone *and* HP-TCP)	21.15 %
GSP2 (UDP)	(UPnP *or* (Full-Cone *and* HP-UDP))	57.66 %
GSP4 (TCP)	(UPnP *or* (Full-Cone *and* HP-TCP))	50.45 %
SPPS1 (UDP)	(HP-UDP)	92.31 %
SPPS2 (TCP)	(HP-TCP)	66.35 %
SPPS3 (TCP)	(HP-TCP *or* HP-UDP)	96.15 %
SPPS4 (UDP)	(UPnP *or* HP-UDP)	92.31 %
SPPS5 (TCP)	(UPnP *or* HP-TCP)	75.96 %
SPPS6 (TCP)	(UPnP *or* HP-TCP *or* HP-UDP)	96.15 %
SSP1 (UDP)	(Restricted-NAT *and* HP-UDP)	49.04 %
SSP2 (TCP)	(Restricted-NAT *and* HP-TCP)	35.58 %

The results of the individual techniques for TCP show that no general solution for TCP-Hole-Punching exists. Therefore, building a framework that automatically detects and applies a working candidate is the most promising way for providing a NAT-Traversal service to applications. Finally, we assumed for all tests that if the NAT-Tester is able to connect to the Test-Server, it is also able to connect to an external relay. Therefore, *100%* of the tested NATs support NAT-Traversal using an appropriate data relay such as TURN.

To adapt the test results to our work, we evaluated the success rates of the individual techniques regarding to our defined service categories. Table 4 shows the categories and the conditions that have to be met according to section 5.2. For example, GSP requires the use of UPnP or working Hole-Punching support in combination with a Full-Cone NAT. Therefore, *57.66%* of our tested NATs supported a direct connection for UDP and category GSP (*50.45%* for TCP). In all other cases (the remaining percentages) an external relay has to be used in order 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 *92.31%* of all NATs we were able to establish a direct connection to the host behind the Network Address Translator (*66.35%* for TCP). This rate increased silghtly for TCP (to *75.96%*) when UPnP was an option. The highest success rate for TCP-NAT-Traversal (*96.15%*) was discovered when we also allowed the tunneling of TCP-packets through UDP.

Finally, SSP only allows 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 *49.04%* for TCP and *35.58%* for UDP.

7 Related Work

A number of techniques have been proposed to examine NAT behavior. The IETF has standardized STUN [6], which is widely used by Voice over IP (VoIP) applications and phones. The STUN protocol determines the public transport address assigned for a connection and detects the NAT strategy of the middlebox by using a number of probing messages. The drawback of STUN is that it does not identify NATs allowing direct control (e.g. UPnP). Additionally, STUN does not preserve its gathered topology knowledge for later usage. This is not an issue for VoIP where there is enough time to perform the STUN protocol until the other party answers the telephone, but many other applications may suffer from this delay.

There is an informational IETF draft presenting results of a survey that also utilized STUN to classify 43 Network Address Translators [16]. Francis did a thorough analysis of NAT implementations in [18] and also considered important properties for NAT-Traversal. However, the tool used is very time consuming and is therefore not able to provide a knowledge gathering component for ANTS.

There are already a number of frameworks for NAT-Traversal that can also incorporate knowledge about the NAT behavior. ICE [2] aims to provide a solution flexible enough to work with all network topologies. Whenever a call is established, each phone gathers all possible transport addresses and exchanges them using SIP. Both clients then set up a local STUN-Server and go through the received candidate list. A client tries to connect to each candidate address using STUN binding-requests and responses, resulting in working candidate pairs. ICE describes a general solution for offer/answer protocols, but was mainly designed to provide a NAT-Traversal solution for VoIP. Therefore, it only supports UDP-based candidates. However, there have been studies on extending ICE to work with TCP as well [19]. ICE also requires both peers to have an ICE-implementation running. If one side does not, ICE cannot help. In [18] and [13], the authors propose a new architecture called NUTSS. They describe the integration of a TCP NAT-Traversal method called Simple Traversal of User Datagram Protocol through NATs and TCP too (STUNT). NatTrav, published in [14], exchanges public endpoints and uses TCP-Hole-Punching to actually traverse the NAT. The proprietary Skype[2] VoIP application has a sophisticated probing technique [20] for NAT-Traversal, which is not available to any other application, though. NAT-Traversal is done using an unknown variation of the STUN and TURN protocol. If possible, a Skype node uses Hole-Punching to establish a direct connection. If not, a Super Node acts as a TURN-Server relaying traffic back and forth.

The analysis of the related work shows that there is no single NAT-Traversal technique which is able to handle all situations without having significant drawbacks in others. This is the strong point of our envisioned architecture that can be extended by behavior and control based NAT-Traversal techniques and profit from persistent knowledge about NAT behavior.

8 Conclusion

The NAT-Traversal problem became more and more important with the increasing popularity of applications following the Peer-to-Peer communication paradigm. Existing solutions to the NAT-Traversal problem have drawbacks of only working with certain types of NAT implementations, or lacking support for legacy applications and multiple transport protocols. When analyzing the NAT-Traversal problem more thoroughly, we discovered that the requirements for legacy applications differ significantly, and identified four service categories relevant for NAT-Traversal. The choice of the NAT-Traversal technique depends on the service category an application belongs to. We made a proposal for a knowledge based extensible NAT-Traversal framework that incorporates the NAT behavior and the available NAT-Traversal options in its decision. As a corner stone of the framework, we implemented the NAT-Tester for knowledge gathering. We did a field test using the NAT-Tester to get an overview on behavior of current NAT implementations and about promising NAT-Traversal techniques. TCP Hole-Punching, for instance, yielded worse results than expected. We therefore must integrate alternative methods to support TCP and SCTP. Future work addresses the implementation of the framework and connecting legacy applications to it.

[2] http://www.skype.com

References

1. Srisuresh, P., Holdrege, M.: IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663, Internet Engineering Task Force (August 1999)
2. Rosenberg, J.: Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for the Session Initiation Protocol (SIP). Internet-Draft - work in progress, Internet Engineering Task Force (October 2007)
3. Rosenberg, J., Mahy, R., Matthews, P.: Traversal Using Relays around NAT (TURN). Internet Draft - work in progress, Internet Engineering Task Force (January 2008)
4. Forum, U.: Internet gateway device (IGD) standardized device control protocol (November 2001)
5. Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A., Rayhan., A.: Middlebox communication architecture and framework. RFC 3303, Internet Engineering Task Force (August 2002)
6. Rosenberg, J., Weinberger, J., Huitema, C., Mahy, R.: STUN: Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). RFC 3489, Internet Engineering Task Force (March 2003)
7. F. Audet, E., Jennings, C.: NAT Behavioral Requirements for Unicast UDP. RFC 4787, Internet Engineering Task Force (January 2007)
8. Guha, S., Francis, P.: Characterization and Measurement of TCP Traversal through NATs and Firewalls. In Proceedings of ACM Interet Measurement Conference (IMC), Berkeley, CA (October 2005)
9. Stiemerling, M., Tschofenig, H., Aoun, C., Davies, E.: NAT/Firewall NSIS Signaling Layer Protocol (NSLP). IETF draft - work in progress, Internet Engineering Task Force (October 2006)
10. Cheshire, S., Krochmal, M., Sekar, K.: NAT Port Mapping Protocol (NAT-PMP). Internet Draft, Internet Engineering Task Force (September 2006)
11. Holdrege, M., Srisuresh, P.: Protocol Complications with the IP Network Address Translator. RFC 3027, Internet Engineering Task Force (January 2001)
12. Ford, B., Srisuresh, P., Kegel, D.: Peer-to-Peer Communication Across Network Address Translation. Technical report, Massachusetts Insitute of Technology (2005)
13. Guha, S., Francis, P.: Towards a Secure Internet Architecture Through Signaling. Technical report, Cornell University (2006)
14. Eppinger, J.: TCP Connections for P2P Applications - A Software Approach to Solving the NAT Problem. Technical report, Carnegie Mellon University, Pittsburgh, PA (2005)
15. P2P-SIP IETF Working Group. http://www.p2psip.org
16. Jennings, C.: NAT Classification Test Results. Internet Draft - work in progress, Internet Engineering Task Force (July 2007)
17. Olsson, M.: Extending the FTP "ALG" vulnerability to any FTP client (March 2000)
18. Francis, P., S.Guha, Takeda, Y.: NUTSS: A SIP-based Approach to UDP and TCP Network Connectivity. Technical report, Cornell University, Panasonic Communications (2004)
19. Rosenberg, J.: TCP Candidates with Interactive Connectivity Establishment. Internet Draft - work in progress, Internet Engineering Task Force (November 2007)
20. Baset, S.A., Schulzrinne, H.: An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. Technical report, Columbia University, New York (2004)