A NAT Traversal Mechanism for Peer-To-Peer Networks

Arno Wacker[1], Gregor Schiele[2], Sebastian Holzapfel[1], and Torben Weis[1]

[1]University of Duisburg-Essen
47057 Duisburg, Germany
firstname.lastname@uni-due.de

[2]University of Mannheim
68131 Mannheim, Germany
gregor.schiele@uni-mannheim.de

Abstract

In this demo we present our approach for establishing a communication channel between hosts behind a NAT-based router. To do so, we developed a peer-to-peer based variant of the STUN protocol using so-called superpeers. Using this protocol we determine the used NAT types for the hosts and select a suitable NAT traversal technique dynamically.

1 Introduction

Network address translation (NAT) [3] is a method which allows to connect multiple hosts to the Internet using a single (public) IP address. The basic idea is to let a *NAT-based router* rewrite the address information in outgoing and incoming messages. This can lead to problems when a peer in a peer-to-peer (P2P) network wants to initiate a new communication with another peer that is connected to the Internet via NAT. Without further arrangements, the NAT-based router will deny this communication request and will drop incoming messages. Thus, the peer is unreachable from other peers. To allow incoming messages to reach the peer, a so-called *NAT traversal* approach is required. One solution for this problem is to let users configure the router manually. However, this is very inconvenient for the user and requires administration rights.

In this demo we address automated NAT traversal for P2P networks. We discuss different NAT types and present our current approach. It is able to establish connections between peers independently of the NAT type and is fully decentralized. To show this in our demo we developed a (centralized) monitoring system which provides a global view of the network, the sent messages, and the used method for traversing certain NAT types. This work is done in the context of the *peers@play* project [2], where we are developing concepts and algorithms for massively multiuser virtual environments.

The paper is structured as follows: In Section 2 we describe different NAT types and existing approaches. After that we present in Section 3 our approach. The demonstration setup of our approach is described in Section 4. We conclude the paper with a short summary and future work in Section 5.

2 NAT Traversal

A NAT-based system consists of a number of hosts in a local network and a NAT-based router that connects this network to the Internet. In the local network, hosts use private IP addresses, e.g., in the 192.168.*.* range. A NAT-based router replaces private addresses (including the used ports) with its own public address before forwarding messages to the Internet. When a reply message arrives, the router does the inverse translation , replacing the public destination address with the local host's private address. To do so, it maintains a table containing the mapping of private to public addresses. An entry is added when a message is sent out. If a message arrives for which no mapping exists, the router does not know the recipient and drops it. Hence, to contact a host behind a NAT-based router a new entry must be added into the NAT table of the receiving peer, such that it becomes reachable.

Following [1] we distinguish between four different NAT types. A *full cone* (FC) creates a mapping for an internal host once the host sends out the first message. After that every host can contact the internal host. *Restricted cone* (RC) allows only hosts that have already been contacted by the internal host to send messages to it. *Port restricted cone* (PRC) additionally requires the external host to use the exact same port number as the internal host used to contact it. Finally, *symmetric* (SYM) enhances the security. Other variants do not modify the internally used port number when sending out a message (if possible). SYM randomly chooses externally used port numbers, making guessing the port number more difficult.

To traverse these NAT variants, different approaches exist. The first one is *reversal*. It can be used if the peer ini-

tiating the communication is behind a FC type and the contacted peer uses one of the more restrictive NAT types. In this case, the first peer cannot reach the second one directly. On the other hand, the second peer can initiate the communication, making itself accessible in the process. Thus, both communication partners switch roles and reverse the communication initiation process. If both communication partners use RC or PRC, the so-called *hole punching* approach can be used. To initiate a communication, peer A sends a message to peer B. Since there is no corresponding mapping in the NAT table used by B's router, this message is dropped. However, it created a new entry in the NAT table on A's router. To reach A, B has to send a message to the exact address used in A's first message, including the used port. Although the first message was dropped and thus never reached B, it is able to guess the correct port number, since the NAT types used here assign outgoing port numbers deterministically. SYM assigns port number randomly. Therefore, the port cannot be guessed and hole punching cannot be used efficiently. *Relaying* can be used if both communication partners are behind SYM. Here, all messages are send indirectly over a third relaying peer, which can be contacted by both peers, e.g., because it is connected directly to the Internet. This adds delay and overhead and is only possible if a suitable relay peer can be found.

3 Our Approach

As described before, there are different approaches for NAT traversal being suitable for different NAT types. Our approach for NAT traversal in P2P networks combines these different NAT traversal techniques. It selects the best one for a given situation at runtime. Therefore, we are able to tolerate many different combinations of NAT types while achieving a high degree performance for the resulting communication. To realize this, our approach has two phases: In the first phase, we detect the used NAT type. In the second phase, we choose the best approach and perform the NAT traversal with it. In the following we describe each of these phases in more detail.

We assume that all peers have already joined the P2P overlay network. The overlay provides multi hop routing of messages between peers and offers a distributed hash table (DHT) interface to store and retrieve data.

To use the best method in each situation, each peer needs to determine its own NAT type. We use the STUN protocol [1] for this. STUN is a UDP-based protocol, that allows a host to detect how it is connected to the Internet, i.e., if it is located behind a NAT, the type of the used NAT, and the host's external (i.e., Internet-wide) address. The host contacts a so-called public STUN server (located outside the host's local network) , which performs a number of tests to reach the host. It reports the result of these tests back to the host. STUN servers are freely available in the Internet and can be used by anyone. In addition, we set up new STUN servers into the P2P network dynamically to be independent of external servers. Such a dynamically introduced STUN server is formed by two peers which use FC or no NAT at all. We call such peers *superpeers*. We require two superpeers for a STUN server because a STUN server needs two public IP addresses. Each superpeer has one such IP address. To detect a STUN server, both external and P2P network internal ones, each peer contacts the existing overlay network using the DHT interface and requests a list of currently available servers. Each newly formed STUN server registers itself at the DHT using a common key. External servers can be registered by an administrator.

After detecting the NAT type, the peer selects a traversal method according to it. We summarize the method used for each combination of NAT type in Table 1. The NAT type of the peer initiating the communication is given in the leftmost column. For the contacted peer it is shown in the uppermost row. For most of these methods, both communication partners must be synchronized with each other. As an example, if a peer A with a RC type wants to contact a peer B with a PRC, both have to punch a hole into their NAT and must coordinate them with each other. We use the existing P2P overlay network to do so. The necessary messages are sent via multiple hops in the overlay. In case relaying should be used, a suitable relay peer must be selected. This is again done by using a superpeer. Each superpeer registers itself in the DHT.

Table 1. NAT Traversal dep. on NAT types

NAT	FC	RC	PRC	SYM
FC	Direct	Reversal	Reversal	Reversal
RC	Direct	Hole Punch.	Hole Punch.	Hole Punch.
PRC	Direct	Hole Punch.	Hole Punch.	Relaying
SYM	Direct	Hole Punch.	Relaying	Relaying

4 Demonstrator

In the context of this demo we present our prototypical implementation of the above presented approach. We will demonstrate how we traverse different NAT types and show the performance of our system. Our system environment on location will consist of a number of peers executed on laptops that will be connected with different NAT types. In addition, we will connect to multiple remote peers running at the Universities of Mannheim and Duisburg-Essen. While the demo is running, remote peers will join and leave the network regularly to simulate an active system. Visitors can include their own laptop (running Windows) into our system at runtime and connect to other peers. To achieve this

we will provide a freely available software package that can be downloaded via our project webpage.

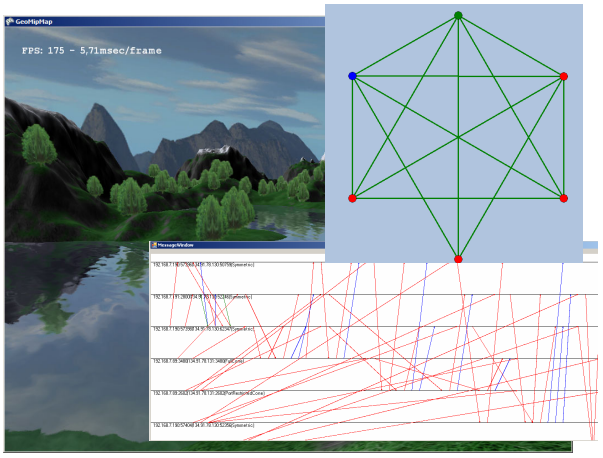

Figure 1. peers@play monitoring and client

Our presented software package consists of a prototypical virtual environment in which users can roam freely and chat with each other. In addition to the 3D visualization of the virtual environment we provide different administration views that show different aspects of our system, as shown in Figure 1. As an example, we will show the message flows between the peers, allowing visitors to follow the used protocols. We also show statistics of the mean delay for establishing a new connection for each NAT type combination.

5 Conclusion & Future Work

In this paper we proposed an approach for NAT traversal in P2P networks. We use STUN to detect a peer's NAT type and select a traversal strategy accordingly, e.g., reversal, or relaying. To detect STUN servers and relaying peers, we use a DHT interface in the P2P network. All necessary coordination between peers is done via the P2P network, too. Thus, no central preconfigured server is necessary for our approach. In future work, we are going to perform a number of experiments to evaluate the system performance. In addition, we plan to extend our approach to traverse firewalls. In many cases, this is already possible with our current approach. However, some firewalls prohibit using UDP. If this is the case, our approach fails.

References

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

[2] G. Schiele, R. Sueselbeck, A. Wacker, J. Haehner, C. Becker, and T. Weis. Requirements of Peer-to-Peer-based Massively Multiplayer Online Gaming. In *Proceedings of the Seventh International Workshop on Global and Peer-to-Peer Computing*, 2007.

[3] P. Srisuresh and K. Egevang. Traditional IP Network Address Translator (Traditional NAT). RFC 3022 (Informational), Jan. 2001.