A New Protocol to Determine the NAT Characteristics of a Host

Sebastian Holzapfel, Matthäus Wander, Arno Wacker, Lorenz Schwittmann, and Torben Weis

University of Duisburg-Essen, Distributed Systems Group,
Bismarckstraße 90, 47057 Duisburg, Germany
{sebastian.holzapfel|matthaeus.wander|arno.wacker|lorenz.schwittmann|torben.weis}@vs.uni-due.de

Abstract—The shortage of IPv4 addresses and the very slow transition to IPv6 leads to well-established pragmatic solutions in the Internet: today many hosts are still using IPv4 and are connected to the Internet over a Network Address Translation router. For many applications, which need inbound connections, like e.g. voice-over-IP or peer-to-peer-based systems it is necessary to determine the characteristics of the surrounding network environment, i.e. the behavior of the used router. In most cases this information is required to successfully establish inbound connections. Therefore, we present in this paper a new protocol to determine the characteristics of the used router. Our protocol is backward compatible to the well known STUN protocol while providing more detailed results. Furthermore, our protocol can be used in a fully decentralized way, i.e. without any centralized servers, making it suitable for pure peer-to-peer-based systems. We complete the presentation of our new protocol with an evaluation through a field experiment.

Keywords—STUN, NAT traversal, Network Address Translation, Peer-to-Peer, P2P

I. INTRODUCTION

Network Address Translation (NAT) [1], [2] is a well-established pragmatic solution to overcome the shortage of IPv4 addresses in the Internet. According to [3], [4] it is currently widely used with more than 70% of all hosts accessing the Internet being situated behind a NAT router. With IPv6 there would be enough IP addresses for every possible host. However, NAT might still be used with IPv6 for topology hiding. Furthermore, the transition to IPv6 is proceeding only very slowly and therefore NAT will remain an established fact for the coming years.

With NAT a single public IP can be shared among multiple hosts behind a NAT router. The hosts behind such a router use private IP addresses, which are replaced with the public address of the router as the packets pass. All corresponding responses are therefore addressed to the public address. Upon arrival of such a response the NAT router replaces the public address with the private internal address and thus allows the hosts in the private network a transparent Internet connectivity.

However, from the Internet only the public IP address of the NAT router is directly reachable. Therefore, the hosts with the private addresses behind the router cannot be addressed directly from other hosts in the Internet. This does not represent a problem for services in the Internet based on a client/server architecture like the WWW or electronic mail. In such cases the hosts in the private network only establish outgoing connections, which are mapped, i.e. translated, by the NAT router. For services, which rely on inbound connections to the hosts behind the NAT router this poses a problem. Modern services like e.g. voice-over-IP (VoIP) or peer-to-peer-based (P2P-based) systems require each host participating in the service to allow for inbound connections. Therefore, so-called NAT traversal mechanisms are required to allow inbound connections for such services.

Several mechanisms for NAT traversal have been proposed in the literature, e.g. [4]–[7]. However, the success of these mechanisms is highly dependent on the existence and behavior, i.e. the characteristics, of the used NAT router. To choose the best suitable mechanism an approach for determining those characteristics is needed.

In this paper we present the Mapping Filtering Behavior (MFB) protocol with the purpose to determine the network characteristics of the used NAT router. Our protocol is based on the existing STUN [8] protocol. We classify the used NAT router by using the mapping and filtering types defined in [9] and detailed in Section II. Additionally, we analyzed the requirements of different NAT traversal mechanisms and extended our protocol with adequate mechanisms. Thus our protocol includes mechanisms for detecting UPnP, availability of the so-called hair-pinning and changes in TCP sequence numbers by the router.

One of the most common applications where inbound connections to hosts behind NAT routers are necessary are P2P-based systems. Therefore we additionally tailored our protocol to cope with the decentralized nature of such systems. In general our protocol requires the MFB service to be provided by some server. However, when used in conjunction with P2P-based systems, some peers can provide this necessary service. By letting the peers provide the server-side service of the MFB protocol we can evade the need for providing publicly available MFB servers. This is simply due to the fact, that the more peers join such a P2P-based system, also more peers are providing the server-side service of the MFB protocol. This feature of our protocol allows for fully decentralized P2P-based systems without the need of any centralized service.
Therefore the contributions of this paper are the requirement analysis of different NAT traversal mechanisms (1), the new MFB protocol for determining the characteristics of the used NAT router (2) and the adaptation of our protocol for purely decentralized systems (3). Additionally, we experimentally evaluated the behavior of different currently used NAT routers as detected with our protocol (4).

This paper is organized as follows: In the next section we present different NAT characteristics used throughout the paper. In Section III we describe different NAT traversal mechanisms and deduce the requirements for our protocol. We then present our approach in Section IV and discuss the adaptation of our protocol for purely decentralized systems in Section V. After that we complete our paper in Section VI with the results from our evaluation. The related work for determining NAT characteristics is discussed in Section VII. Finally, we conclude our paper with a short summary and some thoughts about future work.

II. FUNDAMENTALS

In this section we describe different characteristics [9]–[11] in the behavior of NAT routers. The behavior of NAT routers is not standardized and therefore can differ from router to router. However, the defined characteristics describe current practices and fit to the common behavior of NAT routers.

A. NAT Mapping

In the following we refer to a tuple consisting of an IP address and a port number as IP endpoint. Host A, situated in a private realm, sends packets to a public realm beyond the border of the NAT router. It uses for each outgoing packet the same originating IP endpoint A:a to contact different hosts in the public realm. The NAT mapping describes the translation of the internal IP endpoint A:a to an external IP endpoint and vice versa by the NAT router. In most cases, a NAT router has only one external IP address and therefore the external IP address E will be the same for all created mappings. Thus, the NAT router can only decide which external port is used for a mapping. It can choose a random external port number for each mapping or it can try to preserve the internal port number, if it is not already assigned.

One possible mapping type is to reuse a mapping with the same external IP endpoint E:e, as long as the internal IP endpoint A:a remains constant. The destination IP endpoint is not considered in the mapping procedure, therefore this is called endpoint-independent mapping. In contrast endpoint-dependent mapping uses either the address of the destination host X, the destination port number x, or both of them X:x to establish NAT mappings. If host A uses the same internal IP endpoint A:a but addresses a different destination IP endpoint Y:y, the NAT router will set up a new mapping with a different external IP endpoint E:f. As port number e is already bound, the NAT router can choose a random port number f or it can add or subtract a certain offset, e.g. +1.

In Table I we summarized the possible mapping types which apply to both, TCP and UDP.

Another type, called session-dependent mapping, is an extension of address and port-dependent mapping and applies to TCP only. The NAT router tracks the state of the TCP connections and creates a new mapping with a different external IP endpoint if a connection has been closed and reopened on the same IP endpoint pair (A:a, X:x).

B. NAT Filtering

While mapping describes the behavior of the NAT router for outgoing packets subject to the destination endpoint, filtering describes the behavior for incoming packets subject to the source endpoint. In order to forward incoming packets a mapping must have been set up beforehand by sending an outgoing packet. If there is no mapping on the external IP endpoint addressed by an incoming packet, the packet is filtered as the NAT router can not associate it to any internal IP endpoint.

If the NAT router does not enforce filtering on existing mappings, this behavior is called endpoint-independent filtering. If the NAT router inspects the source address, source port number or both information of incoming packets before forwarding them, it is called endpoint-dependent filtering. The classification shown in Table II is similar to the NAT mapping classification.

Furthermore, a NAT router can silently drop filtered packets or respond with an ICMP error. For TCP packets the NAT router can also respond with another TCP packet with the reset (RST) flag set.

C. Sequence Renumbering

In addition to the alteration of IP endpoints of forwarded packets a NAT router can rewrite the TCP sequence numbers as well. Adding a securely random offset on the sequence numbers of each outgoing TCP connection improves the
protection against IP spoofing from the public realm, if the host in the private realm is using a weak number generator for the initial TCP sequence number.

D. Hair-pinning

Hair-pinning [5] is the ability of a NAT router to forward packets not only between hosts in the public and the private realm but also between hosts within the private realm. This may occur if host A learns of host B’s address from a rendezvous host in the public realm. Though the two hosts could communicate directly, host A is not aware of host B’s internal IP endpoint and therefore addresses the external IP endpoint in the NAT router. If the NAT router does not support hair-pinning, host A will not be able to contact host B unless it learns its internal IP endpoint by other measures.

III. REQUIREMENTS

In this section we describe the required information to decide how a session between two hosts can be established. When speaking about sessions, we refer to both, establishing TCP connections and UDP communication flows. Though the latter was not designed for sessions, UDP NAT mappings essentially represent sessions. If a NAT router prohibits a direct session establishment, there are several NAT traversal mechanisms at choice which differ in terms of cost and availability. We aim to choose the least costly mechanism provided that it is operable in the network environments of two hosts seeking to communicate with each other. Attempting to perform a NAT traversal mechanism that is due to fail anyway would waste resources and time until the session establishment has succeeded or at least has stopped gracefully.

The Internet Gateway Device (IGD) protocol, part of the Universal Plug and Play (UPnP) protocol set, enables hosts to programmatically set up new NAT forwarding rules. UPnP is the most effective mechanism compared to other NAT traversal techniques but it is not enabled on all NAT routers, either due to lack of support or due to security concerns. We thus require to test for UPnP.

Once a host behind a NAT router sends a packet to a host in the public realm, the NAT router creates a mapping which may or may not be used for incoming packets from other hosts. This depends on the filtering behavior of the NAT router which is why it shall be determined. An endpoint-independent filtering indicates a nonrestrictive NAT router or no NAT router at all and allows for direct session establishment without NAT traversal techniques.

When a host’s network environment does not allow for incoming sessions, it may still establish direct sessions by using reversal. Reversal resolves the situation when the destination host B does not allow for incoming sessions, but the initiating host A does. Host A asks host B via an out-of-band channel to open a session back to host A.

If none of the two hosts allow for incoming sessions, a session may be established via hole punching. With hole punching both hosts coordinate to send packets to each other to set up matching mappings and thereby to establish a session. Hole punching is a sophisticated NAT traversal technique, whose success depends on the mapping behavior, the predictability of port assignment, as well as the filtering response behavior of the NAT router. Therefore, these information are required to decide whether hole punching is appropriate. A particular TCP hole punching mechanism [6] is based on sending forged TCP packets from a third host. This mechanism does not have any chance of success if the NAT router rewrites TCP sequence numbers, so this shall be determined too.

If none of these NAT traversal mechanisms are appropriate, hosts may exchange messages by relaying them over a well-known third host, which is directly reachable. The two hosts can use an out-of-band channel to agree on the relaying host. Relaying works independently of the NAT router behavior of host A and B but is also the most expensive mechanism because of the redirection over an otherwise uninvolved host.

The behavior of a NAT router may differ between TCP and UDP. This affects, apart from the TCP specific RST response behavior and sequence renumbering, especially the mapping and filtering behavior. We require to determine the information mentioned above for both protocols in separate tests. Additionally, we also require to determine whether the NAT router supports hair-pinning. Though not a NAT traversal mechanism, the lack of hair-pinning may demand a special handling in the application, e.g. propagating internal IP endpoints to other hosts.

Summarizing the identified requirements, the protocol shall determine the following information:

R1: UPnP available
R2: Filtering behavior
R3: Mapping behavior
R4: Predictability of port assignment
R5: Filtering response behavior
R6: TCP sequence numbers
R7: Separate TCP and UDP results
R8: Hair-pinning support

IV. OUR APPROACH

We now describe a novel protocol called MFB which meets the requirements presented in the previous section. An MFB client looking to determine its network environment communicates with an MFB server using a request-response scheme. The server’s purpose is to aid the clients in gathering the necessary information. An MFB server is required to listen on two public IP addresses (X, Y), each with two ports (x, y) for both, TCP and UDP. MFB servers are stateless.
A client looking to identify its network environment proceeds through the three phases of the protocol, as shown in Figure 1. In the first phase it checks the availability of UPnP. If a UPnP compatible NAT router is available, the client can use it to add a NAT mapping explicitly and the protocol is finished. Otherwise, the client proceeds to the second phase of the protocol, for which it needs to know the public IP endpoints of an MFB server. The second phase comprises the determination of the filtering and response behavior as well as the TCP sequence number test. In the third phase the client determines its mapping behavior and checks if the NAT router support hair-pinning. In the following we describe the messages used in our protocol as well as the three phases in more detail.

A. Messages

We chose to base the message format of the MFB protocol on the STUN protocol specified in [8], to which we will refer to in the following as classic STUN. The MFB client is compatible with both, MFB and classic STUN servers. By utilizing public classic STUN servers, which are still in operation on the Internet, we can execute a limited version of the protocol, even if no MFB servers are available. All messages consist of a 20 byte header including a 2 byte message type, 2 byte message length and 16 bytes transaction ID. The MFB protocol uses two different message types: the client sends a BindingRequest and the server answers with a BindingResponse.

The message header is followed by a variable number of attributes. Each attribute consists of a 2 byte attribute type, 2 byte attribute length, and an arbitrary value, as shown in Figure 2. Attributes are by definition either required or optional ones. We use the same set of required attributes as defined in classic STUN [8] and define new ones as optional. Classic STUN servers ignore optional attributes unknown to them. The BindingRequest message consists of the following attributes:

- **Change_Request** (required): The value of this attribute comprises the two flags ChangeIP and ChangePort. These flags indicate whether the server should reply from the same IP address and port number or different ones.
- **FirewallResponse_Request** (optional): This newly defined attribute instructs the server to detect the response type of closed respectively filtered ports.
- **SequenceNumber_Request** (optional): This attribute instructs the server to include the TCP sequence number as seen in the client’s request.

The BindingResponse message sent by the MFB server consists of the following attributes:

- **Mapped_Address** (required): This attribute reflects the client’s external IP endpoint as seen by the server.
- **Changed_Address** (required): This attribute contains the server’s alternative IP endpoint which is used when the client asks for ChangeIP or ChangePort.
- **Source_Address** (required): This attribute indicates the server’s source IP endpoint used for sending this message. It may be used by the client to detect whether the server is located behind a NAT itself, which may falsify the test results.
- **FirewallResponse** (optional): This newly defined attribute contains the response test result (drop, ICMP reject, TCP RST).
- **SequenceNumber** (optional): This attribute comprises the TCP sequence number in the client’s request as seen by the server.

B. First phase

The first phase consists of the UPnP availability test. The client sends a UDP multicast datagram to port 1900. If there is a NAT router with UPnP support in the local network, it replies with a message containing a hyperlink to an XML document. This document contains information about all provided services. The availability of the service AddPortMapping shows whether UPnP port forwarding is possible.
C. Second phase

During the second phase of the protocol the client determines the filtering and response behavior as well as the TCP sequence number modification. To do so, the client creates two listening sockets, one for TCP and one for UDP. The client sends four BindingRequest messages to an MFB server using the same local source ports as the listening sockets, each with UDP and TCP in parallel. If the client does not know any MFB server, it sends the same requests to a classic STUN server. As classic STUN does not support TCP, the client will in this case not be able to establish TCP connections. Hence, the client is not able to determine the TCP characteristics of its NAT router.

In order to test the four possible filtering types, each BindingRequest contains a different combination of the ChangeIP and ChangePort flags. Therefore, the server uses a different originating IP endpoint for each response. The client waits until it either receives four responses or a timeout occurs. As we show in Table III, the client can derive its filtering type subject to the number of responses received. These apply to both, TCP and UDP. For TCP though, it is required to establish a connection for each response being sent by the server. The initial connection established by the client is used to send the four requests as well as to receive the first response. For the remaining three responses originating from different IP endpoints the server attempts to establish new TCP connections to the client’s IP endpoint.

The initial TCP connection is also used for the TCP sequence number test. The client records the sequence number of the outgoing TCP segment containing the BindingRequest and also includes the SequenceNumber_Request attribute. The server in turn captures the sequence numbers of the incoming TCP segments and returns the value received in the SequenceNumber attribute in the response. By comparing the two values the client learns whether its NAT router changes the sequence numbers. As monitoring sequence numbers usually requires administrator rights under most operating systems, this test may not be available in all cases.

When the client includes the FirewallResponse_Request attribute in any of the requests, the MFB server performs the response behavior test. It sends, depending on the client’s request, some UDP datagrams to the client or attempts to open some TCP connections on randomly chosen ports and waits for replies. As long as the client’s external IP endpoint is not open by coincidence, the server will see whether the client’s NAT router drops filtered packets silently, replies with an ICMP error or with a TCP reset. The MFB server includes the result as value in the FirewallResponse attribute.

At the end of this phase, the client has determined the filtering, response and sequence number behavior. Additionally, the client has determined its external IP endpoint E:e by reading the Mapped_Address attribute.

D. Third phase

During the third phase the client determines the mapping and port assignment behavior and the support for hairpinning of the NAT router. The client sends new BindingRequest messages to the server’s IP endpoints X:x, X:y, Y:y. Again, the client uses the same local source IP endpoint as in the second phase. Unlike in the filtering test, the client does not request ChangeIP or ChangePort from the server to determine the mapping behavior. Thus, the server will use the same IP endpoints for the responses on which the requests have been received and thus all three responses are expected to arrive at the client.

The client determines the mapping behavior subject to the Mapped_Address attributes received in the three responses, as we show in Table IV. If, for example, the mapped addresses of all three responses are equal, then the NAT router uses an endpoint-independent mapping behavior. This applies to UDP as well as to TCP. To identify a session-dependent mapping for TCP, the client needs to reopen the connection to X:x before sending the request. Then, it can compare the mapped address of the response from X:x with the already known external IP endpoint E:e from the previous connection gathered in phase two. If the NAT router uses a different mapping behavior than endpoint-independent, the client determines whether the NAT router uses a predictable port assignment. This happens by checking whether the external ports assigned by the NAT router are located within a small interval (e.g., 10).

In parallel to the mapping behavior test the client also checks whether its NAT router supports hairpinning. For this purpose it needs again its external IP endpoint E:e gathered in phase two. The client attempts to transmit a packet to E:e by using a different local source port as for the previous tests. The NAT router supports hairpinning if the client is able to establish a TCP connection or to send a UDP datagram to itself using the public IP endpoint.

<table>
<thead>
<tr>
<th>Responses received</th>
<th>Filtering behavior</th>
<th>Mapped addresses in responses</th>
<th>Mapping behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response from X:x</td>
<td>Address and port-dependent</td>
<td>MA(X:x) = MA(Y:x) ≠ MA(Y:y)</td>
<td>Endpoint-independent</td>
</tr>
<tr>
<td>Response from X:x and X:y</td>
<td>Address-dependent</td>
<td>MA(X:x) ≠ MA(Y:x) ≠ MA(Y:y)</td>
<td>Address-dependent</td>
</tr>
<tr>
<td>Response from X:x and Y:x</td>
<td>Port-dependent</td>
<td>MA(X:x) ≠ MA(Y:x) ≠ MA(Y:y)</td>
<td>Port-dependent</td>
</tr>
<tr>
<td>Response from X:x, X:y, Y:y</td>
<td>Endpoint-independent</td>
<td>MA(X:x) ≠ E:e</td>
<td>Session-dependent</td>
</tr>
</tbody>
</table>

Table III
FILTERING TYPE IDENTIFICATION SUBJECT TO RESPONSES RECEIVED

Table IV
MAPPING TYPE IDENTIFICATION SUBJECT TO RESPONSES RECEIVED
At the end of this phase the client determined the mapping and port assignment behavior of its NAT router and the support for hair-pinning.

V. P2P ADAPTATION

In this section we describe how to integrate the MFB protocol into a P2P-based system. The goal is to let the participating hosts (peers) provide the MFB service and thereby to diminish the need for dedicated MFB server hosts. We face three challenges when attempting to run the MFB service on peers. First, a joining peer needs to know the IP endpoints of the active MFB peers in order to determine the characteristics of its NAT router. This is virtually the same challenge as the bootstrapping in P2P-based systems, i.e., finding the addresses of already joined peers. Thus, we can use well-established bootstrapping mechanisms to disseminate the addresses of MFB peers (e.g., [12]).

Second, peers may be situated behind a NAT router without allowing incoming connections. We require for an MFB peer to have two external open ports, either set up by manual port forwarding, UPhnP or by an endpoint-independent mapping and filtering behavior with port preservation. This reduces the number of peers coming into question as MFB peer. However, as there are essential services in a P2P-based system requiring publicly reachable peers anyway (e.g., bootstrapping), we can assume that this requirement is met by part of the peers in a functioning system.

The third challenge is that peers rarely have more than one public IP address. Therefore the MFB peer cannot respond to BindingRequest messages with the ChangeIP flag set. To handle this circumstance, we can take advantage of the messaging mechanism of the P2P-based system. The MFB peer can reset the ChangeIP flag, add the requester’s IP endpoint and forward the message to any of its neighbor peers. The neighbor peer receives the forwarded request and can send the BindingResponse originating from a different IP endpoint than the MFB peer.

This approach has some conveniences: The neighbor peer being chosen does not need not to be a publicly reachable peer in order to send the outgoing BindingResponse. As the neighbor peer is already integrated in the P2P-based system, e.g., via NAT traversal, it can receive messages from the MFB peer by using P2P-internal messaging. As any alive neighbor peer of the MFB peer is able to send BindingResponse messages, the MFB peer does not need to actively look for a second MFB peer. This saves maintenance cost in the highly dynamic scenario of a P2P-based system.

There is a possible limitation in the detection of the filtering behavior. If the neighbor peer sending the responses from the alternative IP endpoint is behind a NAT router which does not support port preservation, we cannot ensure that the same external port is used. Thereby, the client may miss the response and erroneously detect a port-dependent filtering type as address and port-dependent filtering. To avoid this, the MFB peer could only consider peers known to preserve the port. However, as we will show in the evaluation in Section VI, port-dependent filtering is of minor practical relevance. Thus, this limitation may be neglected in practice.

VI. EVALUATION

In this section we discuss the results of our experimental evaluation. We implemented the components, i.e., the server and client of our MFB protocol using .NET and C#. Additionally, we used WinPcap [13] for raw access to the network device since .NET does not allow sending and receiving of all packet types, e.g., the ones needed for the firewall response test.

To perform our test, we set up the MFB server with two publicly reachable IP addresses. On each address the MFB server opened two ports, resulting in four publicly reachable IP endpoints, which had been hard-coded into the MFB client. We then distributed the client software to volunteer participants. To detect and collect the NAT characteristics of the routers used by our volunteer participants the client executes our protocol and sends the result back to a database server. In total 40 unique tests were performed by our volunteer participants. In the following we describe the aggregated results of these tests.

In Table V we summarize the mapping behavior of all tested NAT routers. One can observe that the majority of the tested routers exhibit an endpoint-independent mapping behavior. This is in line with similar results presented by Guha et al. in 2005 [9]. Furthermore, the behavior is NAT traversal friendly, since hole punching with this type of endpoint is fairly easy to perform.

Another interesting observation is, that 17.5% of the NAT routers show a different behavior for TCP than for UDP. In general, TCP is more restrictive than UDP.

We summarize the filtering behavior of the tested routers in Table VI. In general the filtering behavior is more restrictive than the mapping behavior shown in Table V:
most routers, i.e. 90% for UDP and 100% for TCP exhibit an address and port-dependent behavior. One router in our test uses address and port-dependent mapping, but endpoint-independent filtering. A more restrictive mapping than filtering is unusual but does not restrain the ability for successful NAT traversal. The filtering behavior is the relevant parameter which dictates the effort needed for a successful inbound connection.

Our results for the router response behavior are summarized in Table VII. The majority of all tested routers silently drop any inbound packets without a corresponding mapping which is the favorable behavior for successful hole punching.

In Table VIII we summarize our results for additional characteristics of the tested routers. We observed, that the majority of routers use a predictable port assignment which is favorable for hole punching. One router uses an assignment which appears to be random and which makes hole punching practically impossible.

In 2005 Ford pointed out the importance of hair-pinning for NAT traversal, especially when there are more levels of NATs involved [5]. However, even after 5 years, only very few routers support hair-pinning in practice. We found one router supporting hair-pinning for both, TCP and UDP. Three more routers support hair-pinning for UDP but not for TCP. These three routers use an endpoint-independent filtering for UDP, but an address and port-dependent filtering for TCP, which suggests the assumption that the hair-pinning support for these routers has been implemented by coincidence rather than on purpose.

None of the routers tested in our experiment altered the TCP sequence numbers. As described in Section III the alteration of TCP sequence numbers renders a particular TCP hole punching mechanism unusable. Therefore, the exhibited behavior is friendly for TCP hole punching.

The result of the UPnP test shows that 10% of the tested devices support UPnP port forwarding and have this feature enabled.

In summary we can deduce from our field experiment that the majority of the routers used today can be used in conjunction with NAT traversal mechanisms that are more efficient than relaying, e.g. hole punching.

### VII. RELATED WORK

As already mentioned, we refer to the ‘Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)’ protocol presented by Rosenberg et al. in [8] as classic STUN. It is the first lightweight client/server protocol to determine the presence and behavior of NAT routers. In contrast to our approach, classic STUN differentiates only between four different NAT types, i.e. FullCone, RestrictedCone, PortRestrictedCone and Symmetric NAT. This classification was found to be too restrictive, since there may be NAT routers that do not fit into these given types [14]. The restriction of classic STUN comes from the fact, that it uses only four different NAT types, whereas the mapping and filtering characteristics as described in [14] and summarized in Section II yield in 16 or 20 distinguishable types, depending whether UDP or TCP is considered. Furthermore, the protocol is based completely on UDP. Therefore no information about the TCP behavior of a router can be determined. When analyzing classic STUN with respect to our requirements, we find, that it fulfills none of them. There are furthermore numerous other publications for determining the NAT behavior and providing NAT traversal, which all rely on the classic STUN classification. All of these approaches cannot fulfill our requirements R2, R3 and R7.

One notable publication which also relies on the classic STUN classification is STUNT by Guha et al. [15]. STUNT extends the classical STUN protocol with support for TCP, therefore fulfilling our requirement R7. However, being based on the classical STUN classification, R2 and R3 cannot be fulfilled.

The classic STUN protocol was obsoleted by [14], which redefines the acronym STUN as ‘Session Traversal Utilities for NAT’. The new specifications replaced the classic NAT classification with new definitions of mapping an filtering behavior. Additionally, the detection algorithm was reduced to a simple mechanism to determine the public IP endpoint used to communicate with the STUN server. We based our approach on the STUN classification of mapping and filtering. However, in contrast to our approach, STUN does not provide the means for determining them. Therefore, STUN does not fulfill any of our requirements.

### Table VII

<table>
<thead>
<tr>
<th>Response behavior</th>
<th>UDP</th>
<th>TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drop</td>
<td>72.5%</td>
<td>80%</td>
</tr>
<tr>
<td>TCP RST</td>
<td>-</td>
<td>5%</td>
</tr>
<tr>
<td>ICMP</td>
<td>27.5%</td>
<td>15%</td>
</tr>
</tbody>
</table>

**Table VII**

**RESPONSE BEHAVIOR FOR UDP AND TCP**

### Table VIII

<table>
<thead>
<tr>
<th>Predictable port assignment</th>
<th>UDP</th>
<th>TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hair-pinning</td>
<td>97.5%</td>
<td>97.5%</td>
</tr>
<tr>
<td>TCP sequence number alterations</td>
<td>10%</td>
<td>2.5%</td>
</tr>
<tr>
<td>UPnP support</td>
<td>-</td>
<td>0%</td>
</tr>
</tbody>
</table>

**Table VIII**

**MISC CHARACTERISTICS FOR UDP AND TCP**

### Table IX

<table>
<thead>
<tr>
<th>Related work</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
<th>R6</th>
<th>R7</th>
<th>R8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classic STUN</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>STUNT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>STUN</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>NAT B.D.</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MFB</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Table IX**

**REQUIREMENTS**
MacDonald and Loweckamp presented in [16] a NAT Behavior Discovery Using STUN (NAT B.D.). They adapted the STUN protocol to detect the mapping and filtering characteristics given in [14] and discussed in Section II. Additionally, they extended the protocol with support for TCP and hair-pinning detection. To do so, they used different attributes from the existing STUN protocols and additionally introduced new mandatory ones. The drawback of this approach is, that their protocol is incompatible with existing STUN servers. Additionally, they do not fulfill our requirements R1, R4, R5 and R6.

To conclude this section we compare in Table VII the approaches discussed above with the requirements defined in Section III.

VIII. CONCLUSION

In this paper we presented our new MFB protocol to enable hosts to determine their NAT router characteristics like the mapping and filtering behavior. The information provided by our protocol fulfill the requirements which we derived from analyzing existing NAT traversal mechanisms. We decided to base our protocol on the existing STUN protocol to reuse publicly available STUN servers. Additionally, we adapted our protocol for purely decentralized P2P-based systems since those systems are one of the main application domains for our protocol. As a proof of concept we implemented our protocol and used it in a field experiment. We observed that hole punching can be performed with the majority of the routers tested, whereas the behavior for UDP is slightly more favorable than for TCP.

As future work we plan to practically evaluate the probability to actually succeed with NAT traversal after the NAT characteristics have been determined with our protocol and a traversal mechanism has been chosen. Depending on the outcome we will examine further parameters which may improve the decision making about the most suitable mechanism, e.g. time-out behavior and TCP state dependencies for incoming and outgoing packets.

REFERENCES


