Network Working Group
Request for Comments: 5128
Category: Informational

P. Srisuresh
Kazeon Systems
B. Ford
M.I.T.
D. Kegel
kegel.com

March 2008

# State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs)

## Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

## Abstract

This memo documents the various methods known to be in use by applications to establish direct communication in the presence of Network Address Translators (NATs) at the current time. Although this memo is intended to be mainly descriptive, the Security Considerations section makes some purely advisory recommendations about how to deal with security vulnerabilities the applications could inadvertently create when using the methods described. This memo covers NAT traversal approaches used by both TCP- and UDP-based applications. This memo is not an endorsement of the methods described, but merely an attempt to capture them in a document.

Table of Contents

# 1. Introduction and Scope

The present-day Internet has seen ubiquitous deployment of Network Address Translators (NATs). There are a variety of NAT devices and a variety of network topologies utilizing NAT devices in deployments. The asymmetric addressing and connectivity regimes established by these NAT devices have created unique problems for peer-to-peer (P2P) applications and protocols, such as teleconferencing and multiplayer online gaming. These issues are likely to persist even into the IPv6 world. During the transition to IPv6, some form of NAT may be required to enable IPv4-only nodes to communicate with IPv6-only nodes [NAT-PT], although the appropriate protocols and guidelines for this use of NAT are still unresolved [NAT-PT-HIST]. Even a future "pure IPv6 world" may still include firewalls, which employ similar filtering behavior of NATs but without the address translation [V6-CPE-SEC]. The filtering behavior interferes with the functioning of P2P applications. For this reason, IPv6 applications that use the techniques described in this document for NAT traversal may also work with some firewalls that have filtering behavior similar to NATs.

Currently deployed NAT devices are designed primarily around the client/server paradigm, in which relatively anonymous client machines inside a private network initiate connections to public servers with stable IP addresses and DNS names. NAT devices encountered en route provide dynamic address assignment for the client machines. The illusion of anonymity (private IP addresses) and inaccessibility of the internal hosts behind a NAT device is not a problem for applications such as Web browsers, which only need to initiate outgoing connections. This illusion of anonymity and inaccessibility is sometimes perceived as a privacy benefit. As noted in Section 2.2 of [RFC4941], this perceived privacy may be illusory in a majority of cases utilizing Small-Office-Home-Office (SOHO) NATs.

In the peer-to-peer paradigm, Internet hosts that would normally be considered "clients" not only initiate sessions to peer nodes, but also accept sessions initiated by peer nodes. The initiator and the responder might lie behind different NAT devices with neither endpoint having a permanent IP address or other form of public network presence. A common online gaming architecture, for example, involves all participating application hosts contacting a publicly addressable rendezvous server for registering themselves and discovering peer hosts. Subsequent to the communication with the rendezvous server, the hosts establish direct connections with each other for fast and efficient propagation of updates during game play. Similarly, a file sharing application might contact a well-known rendezvous server for resource discovery or searching, but establish direct connections with peer hosts for data transfer. NAT devices create problems for peer-to-peer connections because hosts behind a

NAT device normally have no permanently visible public ports on the Internet to which incoming TCP or UDP connections from other peers can be directed. RFC 3235 [NAT-APPL] briefly addresses this issue.

NAT traversal strategies that involve explicit signaling between applications and NAT devices, namely [NAT-PMP], [NSIS-NSLP], [SOCKS], [RSIP], [MIDCOM], and [UPNP] are out of the scope of this document. These techniques, if available, are a complement to the techniques described in the document. [UNSAF] is in scope.

In this document, we summarize the currently known methods by which applications work around the presence of NAT devices, without directly altering the NAT devices. The techniques described predate BEHAVE documents ([BEH-UDP], [BEH-TCP], and [BEH-ICMP]). The scope of the document is restricted to describing currently known techniques used to establish 2-way communication between endpoints of an application. Discussion of timeouts, RST processing, keepalives, and so forth that concern a running session are outside the scope of this document. The scope is also restricted to describing techniques for TCP- and UDP-based applications. It is not the objective of this document to provide solutions to NAT traversal problems for applications in general [BEH-APP] or to a specific class of applications [ICE].

## 2. Terminology and Conventions Used

In this document, the IP addresses 192.0.2.1, 192.0.2.128, and 192.0.2.254 are used as example public IP addresses [RFC3330]. Although these addresses are all from the same /24 network, this is a limitation of the example addresses available in [RFC3330]. In practice, these addresses would be on different networks. As for the notation for ports usage, all clients use ports in the range of 1-2000 and servers use ports in the range of 20000-21000. NAT devices use ports 30000 and above for endpoint mapping.

Readers are urged to refer to [NAT-TERM] for information on NAT taxonomy and terminology. Unless prefixed with a NAT type or explicitly stated otherwise, the term NAT, used throughout this document, refers to Traditional NAT [NAT-TRAD]. Traditional NAT has two variations, namely, Basic NAT and Network Address Port Translator (NAPT). Of these, NAPT is by far the most commonly deployed NAT device. NAPT allows multiple private hosts to share a single public IP address simultaneously.

An issue of relevance to P2P applications is how the NAT behaves when an internal host initiates multiple simultaneous sessions from a single endpoint (private IP, private port) to multiple distinct endpoints on the external network.

[STUN] further classifies NAT implementations using the terms "Full Cone", "Restricted Cone", "Port Restricted Cone", and "Symmetric". Unfortunately, this terminology has been the source of much confusion. For this reason, this document adapts terminology from [BEH-UDP] to distinguish between NAT implementations.

Listed below are terms used throughout this document.

### 2.1. Endpoint

An endpoint is a session-specific tuple on an end host. An endpoint may be represented differently for each IP protocol. For example, a UDP or TCP session endpoint is represented as a tuple of (IP address, UDP/TCP port).

### 2.2. Endpoint Mapping

When a host in a private realm initiates an outgoing session to a host in the public realm through a NAT device, the NAT device assigns a public endpoint to translate the private endpoint so that subsequent response packets from the external host can be received by the NAT, translated, and forwarded to the private endpoint. The assignment by the NAT device to translate a private endpoint to a public endpoint and vice versa is called Endpoint Mapping. NAT uses Endpoint Mapping to perform translation for the duration of the session.

### 2.3. Endpoint-Independent Mapping

"Endpoint-Independent Mapping" is defined in [BEH-UDP] as follows:

> The NAT reuses the port mapping for subsequent packets sent from the same internal IP address and port (X:x) to any external IP address and port.

### 2.4. Endpoint-Dependent Mapping

"Endpoint-Dependent Mapping" refers to the combination of "Address-Dependent Mapping" and "Address and Port-Dependent Mapping" as defined in [BEH-UDP]:

Address-Dependent Mapping

> The NAT reuses the port mapping for subsequent packets sent from the same internal IP address and port (X:x) to the same external IP address, regardless of the external port.

Address and Port-Dependent Mapping

The NAT reuses the port mapping for subsequent packets sent from the same internal IP address and port (X:x) to the same external IP address and port while the mapping is still active.

## 2.5. Endpoint-Independent Filtering

"Endpoint-Independent Filtering" is defined in [BEH-UDP] as follows:

The NAT filters out only packets not destined to the internal address and port X:x, regardless of the external IP address and port source (Z:z). The NAT forwards any packets destined to X:x. In other words, sending packets from the internal side of the NAT to any external IP address is sufficient to allow any packets back to the internal endpoint.

A NAT device employing the combination of "Endpoint-Independent Mapping" and "Endpoint-Independent Filtering" will accept incoming traffic to a mapped public port from ANY external endpoint on the public network.

## 2.6. Endpoint-Dependent Filtering

"Endpoint-Dependent Filtering" refers to the combination of "Address-Dependent Filtering" and "Address and Port-Dependent Filtering" as defined in [BEH-UDP].

Address-Dependent Filtering

The NAT filters out packets not destined to the internal address X:x. Additionally, the NAT will filter out packets from Y:y destined for the internal endpoint X:x if X:x has not sent packets to Y:any previously (independently of the port used by Y). In other words, for receiving packets from a specific external endpoint, it is necessary for the internal endpoint to send packets first to that specific external endpoint’s IP address.

Address and Port-Dependent Filtering

The NAT filters out packets not destined for the internal address X:x. Additionally, the NAT will filter out packets from Y:y destined for the internal endpoint X:x if X:x has not sent packets to Y:y previously. In other words, for receiving packets from a specific external endpoint, it is necessary for the internal endpoint to send packets first to that external endpoint’s IP address and port.

A NAT device employing "Endpoint-Dependent Filtering" will accept incoming traffic to a mapped public port from only a restricted set of external endpoints on the public network.

## 2.7. P2P Application

A P2P application is an application that uses the same endpoint to initiate outgoing sessions to peering hosts as well as accept incoming sessions from peering hosts. A P2P application may use multiple endpoints for peer-to-peer communication.

## 2.8. NAT-Friendly P2P Application

A NAT-friendly P2P application is a P2P application that is designed to work effectively even as peering nodes are located in distinct IP address realms, connected by one or more NATs.

One common way P2P applications establish peering sessions and remain NAT-friendly is by using a publicly addressable rendezvous server for registration and peer discovery purposes.

## 2.9. Endpoint-Independent Mapping NAT (EIM-NAT)

An Endpoint-Independent Mapping NAT (EIM-NAT, for short) is a NAT device employing Endpoint-Independent Mapping. An EIM-NAT can have any type of filtering behavior. BEHAVE-compliant NAT devices are good examples of EIM-NAT devices. A NAT device employing Address-Dependent Mapping is an example of a NAT device that is not EIM-NAT.

## 2.10. Hairpinning

Hairpinning is defined in [BEH-UDP] as follows:

> If two hosts (called X1 and X2) are behind the same NAT and exchanging traffic, the NAT may allocate an address on the outside of the NAT for X2, called X2':x2'. If X1 sends traffic to X2':x2', it goes to the NAT, which must relay the traffic from X1 to X2. This is referred to as hairpinning.

Not all currently deployed NATs support hairpinning.

# 3. Techniques Used by P2P Applications to Traverse NATs

This section reviews in detail the currently known techniques for implementing peer-to-peer communication over existing NAT devices, from the perspective of the application or protocol designer.

## 3.1. Relaying

The most reliable, but least efficient, method of implementing peer-to-peer communication in the presence of a NAT device is to make the peer-to-peer communication look to the network like client/server communication through relaying. Consider the scenario in figure 1. Two client hosts, A and B, have each initiated TCP or UDP connections to a well-known rendezvous server S. The Rendezvous Server S has a publicly addressable IP address and is used for the purposes of registration, discovery, and relay. Hosts behind NAT register with the server. Peer hosts can discover hosts behind NATs and relay all end-to-end messages using the server. The clients reside on separate private networks, and their respective NAT devices prevent either client from directly initiating a connection to the other.

```
                           Registry, Discovery
                           Combined with Relay
                                 Server S
                            192.0.2.128:20001
                                     |
        +----------------------------+----------------------------+
        | ^ Registry/              ^   ^ Registry/              ^ |
        | | Relay-Req Session(A-S) |   | Relay-Req Session(B-S) | |
        | | 192.0.2.128:20001      |   |  192.0.2.128:20001     | |
        | | 192.0.2.1:62000        |   |  192.0.2.254:31000     | |
        |                                                         |
      +--------------+                                 +--------------+
      | 192.0.2.1    |                                 | 192.0.2.254  |
      |              |                                 |              |
      |    NAT A     |                                 |    NAT B     |
      +--------------+                                 +--------------+
        |                                                         |
        | ^ Registry/              ^   ^ Registry/              ^ |
        | | Relay-Req Session(A-S) |   | Relay-Req Session(B-S) | |
        | |  192.0.2.128:20001     |   |  192.0.2.128:20001     | |
        | |     10.0.0.1:1234      |   |     10.1.1.3:1234      | |
        |                                                         |
     Client A                                                 Client B
     10.0.0.1:1234                                        10.1.1.3:1234
```

Figure 1: Use of a Relay Server to communicate with peers

Instead of attempting a direct connection, the two clients can simply use the server S to relay messages between them. For example, to send a message to client B, client A simply sends the message to server S along its already established client/server connection, and server S then sends the message on to client B using its existing client/server connection with B.

This method has the advantage that it will always work as long as both clients have connectivity to the server. The enroute NAT device is not required to be EIM-NAT. The obvious disadvantages of relaying are that it consumes the server's processing power and network bandwidth, and communication latency between the peering clients is likely to be increased even if the server has sufficient I/O bandwidth and is located correctly topology-wise. The TURN protocol [TURN] defines a method of implementing application agnostic, session-oriented, packet relay in a relatively secure fashion.

### 3.2. Connection Reversal

The following connection reversal technique for a direct communication works only when one of the peers is behind a NAT device and the other is not. For example, consider the scenario in figure 2. Client A is behind a NAT, but client B has a publicly addressable IP address. Rendezvous Server S has a publicly addressable IP address and is used for the purposes of registration and discovery. Hosts behind a NAT register their endpoints with the server. Peer hosts discover endpoints of hosts behind a NAT using the server.

```
                          Registry and Discovery
                                 Server S
                            192.0.2.128:20001
                                     |
        +----------------------------+----------------------------+
        | ^ Registry Session(A-S) ^     ^ Registry Session(B-S) ^ |
        | | 192.0.2.128:20001     |     |  192.0.2.128:20001    | |
        | | 192.0.2.1:62000       |     |  192.0.2.254:1234     | |
        |                                                         |
        | ^ P2P Session (A-B)     ^     |  P2P Session (B-A)    | |
        | | 192.0.2.254:1234      |     |  192.0.2.1:62000      | |
        | | 192.0.2.1:62000       |     v  192.0.2.254:1234     v |
        |                                                         |
      +--------------+                                            |
      | 192.0.2.1    |                                            |
      |              |                                            |
      |    NAT A     |                                            |
      +--------------+                                            |
        |                                                         |
        | ^ Registry Session(A-S) ^                               |
        | |  192.0.2.128:20001    |                               |
        | |     10.0.0.1:1234     |                               |
        |                                                         |
        | ^ P2P Session (A-B)     ^                               |
        | |  192.0.2.254:1234     |                               |
        | |     10.0.0.1:1234     |                               |
        |                                                         |
     Private Client A                                 Public Client B
     10.0.0.1:1234                                    192.0.2.254:1234
```

Figure 2: Connection reversal using Rendezvous server

Client A has private IP address 10.0.0.1, and the application is using TCP port 1234. This client has established a connection with server S at public IP address 192.0.2.128 and port 20001. NAT A has assigned TCP port 62000, at its own public IP address 192.0.2.1, to serve as the temporary public endpoint address for A’s session with S; therefore, server S believes that client A is at IP address 192.0.2.1 using port 62000. Client B, however, has its own permanent IP address, 192.0.2.254, and the application on B is accepting TCP connections at port 1234.

Now suppose client B wishes to establish a direct communication session with client A. B might first attempt to contact client A either at the address client A believes itself to have, namely, 10.0.0.1:1234, or at the address of A as observed by server S, namely, 192.0.2.1:62000. In either case, the connection will fail. In the first case, traffic directed to IP address 10.0.0.1 will simply be dropped by the network because 10.0.0.1 is not a publicly routable IP address. In the second case, the TCP SYN request from B will arrive at NAT A directed to port 62000, but NAT A will reject the connection request because only outgoing connections are allowed.

After attempting and failing to establish a direct connection to A, client B can use server S to relay a request to client A to initiate a "reversed" connection to client B. Client A, upon receiving this relayed request through S, opens a TCP connection to client B at B's public IP address and port number. NAT A allows the connection to proceed because it is originating inside the firewall, and client B can receive the connection because it is not behind a NAT device.

A variety of current peer-to-peer applications implement this technique. Its main limitation, of course, is that it only works so long as only one of the communicating peers is behind a NAT device. If the NAT device is EIM-NAT, the public client can contact external server S to determine the specific public endpoint from which to expect Client-A-originated connection and allow connections from just those endpoints. If the NAT device is EIM-NAT, the public client can contact the external server S to determine the specific public endpoint from which to expect connections originated by client A, and allow connections from just that endpoint. If the NAT device is not EIM-NAT, the public client cannot know the specific public endpoint from which to expect connections originated by client A. In the increasingly common case where both peers can be behind NATs, the Connection Reversal method fails. Connection Reversal is not a general solution to the peer-to-peer connection problem. If neither a "forward" nor a "reverse" connection can be established, applications often fall back to another mechanism such as relaying.

### 3.3. UDP Hole Punching

UDP hole punching relies on the properties of EIM-NATs to allow appropriately designed peer-to-peer applications to "punch holes" through the NAT device(s) enroute and establish direct connectivity with each other, even when both communicating hosts lie behind NAT devices. When one of the hosts is behind a NAT that is not EIM-NAT, the peering host cannot predictably know the mapped endpoint to which to initiate a connection. Further, the application on the host behind non-EIM-NAT would be unable to reuse an already established endpoint mapping for communication with different external destinations, and the hole punching technique would fail.

This technique was mentioned briefly in Section 5.1 of RFC 3027 [NAT-PROT], first described in [KEGEL], and used in some recent protocols [TEREDO, ICE]. Readers may refer to Section 3.4 for details on "TCP hole punching".

We will consider two specific scenarios, and how applications are designed to handle both of them gracefully. In the first situation, representing the common case, two clients desiring direct peer-to-peer communication reside behind two different NATs. In the second, the two clients actually reside behind the same NAT, but do not necessarily know that they do.

### 3.3.1. Peers behind Different NATs

Consider the scenario in figure 3. Clients A and B both have private IP addresses and lie behind different NAT devices. Rendezvous Server S has a publicly addressable IP address and is used for the purposes of registration, discovery, and limited relay. Hosts behind a NAT register their public endpoints with the server. Peer hosts discover the public endpoints of hosts behind a NAT using the server. Unlike in Section 3.1, peer hosts use the server to relay just connection initiation control messages, instead of end-to-end messages.

The peer-to-peer application running on clients A and B use UDP port 1234. The rendezvous server S uses UDP port 20001. A and B have each initiated UDP communication sessions with server S, causing NAT A to assign its own public UDP port 62000 for A’s session with S, and causing NAT B to assign its port 31000 to B’s session with S, respectively.

```
                      Registry and Discovery Combined
                            with Limited Relay
                                 Server S
                              192.0.2.128:20001
                                     |
        +----------------------------+----------------------------+
        | ^ Registry Session(A-S) ^      ^ Registry Session(B-S) ^ |
        | | 192.0.2.128:20001     |      |  192.0.2.128:20001    | |
        | | 192.0.2.1:62000       |      |  192.0.2.254:31000    | |
        |                                                          |
        | ^ P2P Session (A-B)     ^      ^  P2P Session (B-A)    ^ |
        | | 192.0.2.254:31000     |      |  192.0.2.1:62000      | |
        | | 192.0.2.1:62000       |      |  192.0.2.254:31000    | |
        |                                                          |
      +--------------+                                  +--------------+
      | 192.0.2.1    |                                  | 192.0.2.254  |
      |              |                                  |              |
      | EIM-NAT A    |                                  | EIM-NAT B    |
      +--------------+                                  +--------------+
        |                                                          |
        | ^ Registry Session(A-S) ^      ^ Registry Session(B-S) ^ |
        | |  192.0.2.128:20001    |      |  192.0.2.128:20001    | |
        | |     10.0.0.1:1234     |      |     10.1.1.3:1234     | |
        |                                                          |
        | ^ P2P Session (A-B)     ^      ^  P2P Session (B-A)    ^ |
        | |  192.0.2.254:31000    |      |  192.0.2.1:62000      | |
        | |     10.0.0.1:1234     |      |     10.1.1.3:1234     | |
        |                                                          |
     Client A                                                Client B
     10.0.0.1:1234                                      10.1.1.3:1234
```

Figure 3: UDP Hole Punching to set up direct connectivity

Now suppose that client A wants to establish a UDP communication session directly with client B. If A simply starts sending UDP messages to B’s public endpoint 192.0.2.254:31000, then NAT B will typically discard these incoming messages (unless it employs Endpoint-Independent Filtering), because the source address and port number do not match those of S, with which the original outgoing session was established. Similarly, if B simply starts sending UDP messages to A’s public endpoint, then NAT A will typically discard these messages.

Suppose A starts sending UDP messages to B’s public endpoint, and simultaneously relays a request through server S to B, asking B to start sending UDP messages to A’s public endpoint. A’s outgoing messages directed to B’s public endpoint (192.0.2.254:31000) cause EIM-NAT A to open up a new communication session between A’s private endpoint and B’s public endpoint. At the same time, B’s messages to A’s public endpoint (192.0.2.1:62000) cause EIM-NAT B to open up a new communication session between B’s private endpoint and A’s public endpoint. Once the new UDP sessions have been opened up in each direction, clients A and B can communicate with each other directly without further burden on the server S. Server S, which helps with relaying connection initiation requests to peer nodes behind NAT devices, ends up like an "introduction" server to peer hosts.

The UDP hole punching technique has several useful properties. Once a direct peer-to-peer UDP connection has been established between two clients behind NAT devices, either party on that connection can in turn take over the role of "introducer" and help the other party establish peer-to-peer connections with additional peers, minimizing the load on the initial introduction server S. The application does not need to attempt to detect the kind of NAT device it is behind, since the procedure above will establish peer-to-peer communication channels equally well if either or both clients do not happen to be behind a NAT device. The UDP hole punching technique even works automatically with multiple NATs, where one or both clients are distant from the public Internet via two or more levels of address translation.

### 3.3.2. Peers behind the Same NAT

Now consider the scenario in which the two clients (probably unknowingly) happen to reside behind the same EIM-NAT, and are therefore located in the same private IP address space, as in figure 4. A well-known Rendezvous Server S has a publicly addressable IP address and is used for the purposes of registration, discovery, and limited relay. Hosts behind the NAT register with the server. Peer hosts discover hosts behind the NAT using the server and relay messages using the server. Unlike in Section 3.1, peer hosts use the server to relay just control messages, instead of all end-to-end messages.

Client A has established a UDP session with server S, to which the common EIM-NAT has assigned public port number 62000. Client B has similarly established a session with S, to which the EIM-NAT has assigned public port number 62001.

```
                     Registry and Discovery Combined
                           with Limited Relay
                                Server S
                            192.0.2.128:20001
                                    |
         ^ Registry Session(A-S) ^  | ^ Registry Session(B-S) ^
         | 192.0.2.128:20001     |  | |  192.0.2.128:20001    |
         | 192.0.2.1:62000       |  | |  192.0.2.1:62001      |
                                    |
                             +--------------+
                             | 192.0.2.1    |
                             |              |
                             |   EIM-NAT    |
                             +--------------+
                                    |
      +-----------------------------+----------------------------+
      | ^ Registry Session(A-S) ^      ^ Registry Session(B-S) ^ |
      | |  192.0.2.128:20001    |      |  192.0.2.128:20001    | |
      | |     10.0.0.1:1234     |      |     10.1.1.3:1234     | |
      |                                                          |
      | ^ P2P Session-try1(A-B) ^      ^ P2P Session-try1(B-A) ^ |
      | | 192.0.2.1:62001       |      |  192.0.2.1:62000      | |
      | |     10.0.0.1:1234     |      |     10.1.1.3:1234     | |
      |                                                          |
      | ^ P2P Session-try2(A-B) ^      ^ P2P Session-try2(B-A) ^ |
      | |     10.1.1.3:1234     |      |     10.0.0.1:1234     | |
      | |     10.0.0.1:1234     |      |     10.1.1.3:1234     | |
      |                                                          |
   Client A                                                Client B
   10.0.0.1:1234                                      10.1.1.3:1234
```

Figure 4: Use of local and public endpoints to communicate with peers

Suppose that A and B use the UDP hole punching technique as outlined above to establish a communication channel using server S as an introducer. Then A and B will learn each other's public endpoints as observed by server S, and start sending each other messages at those public endpoints. The two clients will be able to communicate with each other this way as long as the NAT allows hosts on the internal network to open translated UDP sessions with other internal hosts and not just with external hosts. This situation is referred to as "Hairpinning", because packets arriving at the NAT from the private network are translated and then looped back to the private network rather than being passed through to the public network.

For example, consider P2P session-try1 above. When A sends a UDP packet to B's public endpoint, the packet initially has a source endpoint of 10.0.0.1:1234 and a destination endpoint of

192.0.2.1:62001. The NAT receives this packet, translates it to have a source endpoint of 192.0.2.1:62000 and a destination endpoint of 10.1.1.3:1234, and then forwards it on to B.

Even if the NAT device supports hairpinning, this translation and forwarding step is clearly unnecessary in this situation, and adds latency to the dialog between A and B, besides burdening the NAT. The solution to this problem is straightforward and is described as follows.

When A and B initially exchange address information through the Rendezvous server S, they include their own IP addresses and port numbers as "observed" by themselves, as well as their public endpoints as observed by S. The clients then simultaneously start sending packets to each other at each of the alternative addresses they know about, and use the first address that leads to successful communication. If the two clients are behind the same NAT, as is the case in figure 4 above, then the packets directed to their private endpoints (as attempted using P2P session-try2) are likely to arrive first, resulting in a direct communication channel not involving the NAT. If the two clients are behind different NATs, then the packets directed to their private endpoints will fail to reach each other at all, but the clients will hopefully establish connectivity using their respective public endpoints. It is important that these packets be authenticated in some way, however, since in the case of different NATs it is entirely possible for A’s messages directed at B’s private endpoint to reach some other, unrelated node on A’s private network, or vice versa.

The [ICE] protocol employs this technique effectively, in that multiple candidate endpoints (both private and public) are communicated between peering end hosts during an offer/answer exchange. Endpoints that offer the most efficient end-to-end connection(s) are selected eventually for end-to-end data transfer.

### 3.3.3. Peers Separated by Multiple NATs

In some topologies involving multiple NAT devices, it is not possible for two clients to establish an "optimal" P2P route between them without specific knowledge of the topology. Consider for example the scenario in figure 5.

```
                     Registry and Discovery Combined
                           with Limited Relay
                                Server S
                           192.0.2.128:20001
                                   |
         ^ Registry Session(A-S) ^ | ^ Registry Session(B-S) ^
         | 192.0.2.128:20001     | | | 192.0.2.128:20001     |
         | 192.0.2.1:62000       | | | 192.0.2.1:62001       |
                                   |
                            +--------------+
                            | 192.0.2.1    |
                            |              |
                            |  EIM-NAT X   |
                            | (Supporting  |
                            | Hairpinning) |
                            +--------------+
                                   |
      +----------------------------+----------------------------+
      | ^ Registry Session(A-S) ^     ^ Registry Session(B-S) ^ |
      | |  192.0.2.128:20001    |     |  192.0.2.128:20001    | |
      | |  192.168.1.1:30000    |     |  192.168.1.2:31000    | |
      |                                                         |
      | ^ P2P Session (A-B)     ^     ^ P2P Session (B-A)     ^ |
      | |  192.0.2.1:62001      |     |  192.0.2.1:62000      | |
      | |  192.168.1.1:30000    |     |  192.168.1.2:31000    | |
      |                                                         |
   +--------------+                                  +--------------+
   | 192.168.1.1  |                                  | 192.168.1.2  |
   |              |                                  |              |
   | EIM-NAT A    |                                  | EIM-NAT B    |
   +--------------+                                  +--------------+
       |                                                        |
       | ^ Registry Session(A-S) ^    ^ Registry Session(B-S) ^ |
       | |  192.0.2.128:20001    |    |  192.0.2.128:20001    | |
       | |     10.0.0.1:1234     |    |     10.1.1.3:1234     | |
       |                                                        |
       | ^ P2P Session (A-B)     ^    ^  P2P Session (B-A)    ^ |
       | |  192.0.2.1:62001      |    |  192.0.2.1:62000      | |
       | |     10.0.0.1:1234     |    |     10.1.1.3:1234     | |
       |                                                        |
   Client A                                                  Client B
   10.0.0.1:1234                                        10.1.1.3:1234
```

Figure 5: Use of Hairpinning in setting up direct communication

Suppose NAT X is an EIM-NAT deployed by a large Internet Service Provider (ISP) to multiplex many customers onto a few public IP addresses, and NATs A and B are small consumer NAT gateways deployed independently by two of the ISP's customers to multiplex their private home networks onto their respective ISP-provided IP addresses. Only server S and NAT X have globally routable IP addresses; the "public" IP addresses used by NAT A and NAT B are actually private to the ISP's addressing realm, while client A's and B's addresses in turn are private to the addressing realms of NATs A and B, respectively. Just as in the previous section, server S is used for the purposes of registration, discovery, and limited relay. Peer hosts use the server to relay connection initiation control messages, instead of all end-to-end messages.

Now suppose clients A and B attempt to establish a direct peer-to-peer UDP connection. The optimal method would be for client A to send messages to client B's public address at NAT B, 192.168.1.2:31000 in the ISP's addressing realm, and for client B to send messages to A's public address at NAT B, namely, 192.168.1.1:30000. Unfortunately, A and B have no way to learn these addresses, because server S only sees the "global" public endpoints of the clients, 192.0.2.1:62000 and 192.0.2.1:62001. Even if A and B had some way to learn these addresses, there is still no guarantee that they would be usable because the address assignments in the ISP's private addressing realm might conflict with unrelated address assignments in the clients' private realms. The clients therefore have no choice but to use their global public endpoints as seen by S for their P2P communication, and rely on NAT X to provide hairpinning.

## 3.4. TCP Hole Punching

In this section, we will discuss the "TCP hole punching" technique used for establishing direct TCP connection between a pair of nodes that are both behind EIM-NAT devices. Just as with UDP hole punching, TCP hole punching relies on the properties of EIM-NATs to allow appropriately designed peer-to-peer applications to "punch holes" through the NAT device and establish direct connectivity with each other, even when both communicating hosts lie behind NAT devices. This technique is also known sometimes as "Simultaneous TCP Open".

Most TCP sessions start with one endpoint sending a SYN packet, to which the other party responds with a SYN-ACK packet. It is permissible, however, for two endpoints to start a TCP session by simultaneously sending each other SYN packets, to which each party subsequently responds with a separate ACK. This procedure is known as "Simultaneous TCP Open" technique and may be found in figure 6 of the original TCP specification ([TCP]). However, "Simultaneous TCP Open" is not implemented correctly on many systems, including NAT devices.

If a NAT device receives a TCP SYN packet from outside the private network attempting to initiate an incoming TCP connection, the NAT device will normally reject the connection attempt by either dropping the SYN packet or sending back a TCP RST (connection reset) packet. In the case of SYN timeout or connection reset, the application endpoint will continue to resend a SYN packet, until the peer does the same from its end.

Let us consider the case where a NAT device supports "Simultaneous TCP Open" sessions. When a SYN packet arrives with source and destination endpoints that correspond to a TCP session that the NAT device believes is already active, then the NAT device would allow the packet to pass through. In particular, if the NAT device has just recently seen and transmitted an outgoing SYN packet with the same address and port numbers, then it will consider the session active and allow the incoming SYN through. If clients A and B can each initiate an outgoing TCP connection with the other client timed so that each client’s outgoing SYN passes through its local NAT device before either SYN reaches the opposite NAT device, then a working peer-to-peer TCP connection will result.

This technique may not always work reliably for the following reason(s). If either node’s SYN packet arrives at the remote NAT device too quickly (before the peering node had a chance to send the SYN packet), then the remote NAT device may either drop the SYN packet or reject the SYN with a RST packet. This could cause the local NAT device in turn to close the new NAT session immediately or initiate end-of-session timeout (refer to Section 2.6 of [NAT-TERM]) so as to close the NAT session at the end of the timeout. Even as both peering nodes simultaneously initiate continued SYN retransmission attempts, some remote NAT devices might not let the incoming SYNs through if the NAT session is in an end-of-session timeout state. This in turn would prevent the TCP connection from being established.

In reality, the majority of NAT devices (more than 50%) support Endpoint-Independent Mapping and do not send ICMP errors or RSTs in response to unsolicited incoming SYNs. As a result, the Simultaneous TCP Open technique does work across NAT devices in the majority of TCP connection attempts ([P2P-NAT], [TCP-CHARACT]).

### 3.5. UDP Port Number Prediction

A variant of the UDP hole punching technique exists that allows peer-to-peer UDP sessions to be created in the presence of some NATs implementing Endpoint-Dependent Mapping. This method is sometimes called the "N+1" technique [BIDIR] and is explored in detail by Takeda [SYM-STUN]. The method works by analyzing the behavior of the

NAT and attempting to predict the public port numbers it will assign to future sessions. The public ports assigned are often predictable because most NATs assign mapping ports in sequence.

Consider the scenario in figure 6. Two clients, A and B, each behind a separate NAT, have established separate UDP connections with rendezvous server S. Rendezvous server S has a publicly addressable IP address and is used for the purposes of registration and discovery. Hosts behind a NAT register their endpoints with the server. Peer hosts discover endpoints of the hosts behind NAT using the server.

```
                           Registry and Discovery
                                  Server S
                              192.0.2.128:20001
                                      |
                                      |
       +------------------------------+----------------------------+
       | ^ Registry Session(A-S) ^     ^ Registry Session(B-S) ^ |
       | | 192.0.2.128:20001     |     |  192.0.2.128:20001    | |
       | | 192.0.2.1:62000       |     |  192.0.2.254:31000    | |
       |                                                          |
       | ^ P2P Session (A-B)     ^     ^  P2P Session (B-A)    ^ |
       | | 192.0.2.254:31001     |     |  192.0.2.1:62001      | |
       | | 192.0.2.1:62001       |     |  192.0.2.254:31001    | |
       |                                                          |
  +----------------------+                       +----------------------+
  | 192.0.2.1            |                       |        192.0.2.254 |
  |                      |                       |                      |
  |    NAT A             |                       |        NAT B         |
  | (Endpoint-Dependent  |                       | (Endpoint-Dependent|
  |  Mapping)            |                       |  Mapping)            |
  +----------------------+                       +----------------------+
       |                                                          |
       | ^ Registry Session(A-S) ^     ^ Registry Session(B-S) ^ |
       | |  192.0.2.128:20001    |     |  192.0.2.128:20001    | |
       | |     10.0.0.1:1234     |     |     10.1.1.3:1234     | |
       |                                                          |
       | ^ P2P Session (A-B)     ^     ^ P2P Session (B-A)     ^ |
       | |  192.0.2.254:31001    |     |  192.0.2.1:62001      | |
       | |     10.0.0.1:1234     |     |     10.1.1.3:1234     | |
       |                                                          |
     Client A                                              Client B
     10.0.0.1:1234                                      10.1.1.3:1234
```

Figure 6: UDP Port Prediction to set up direct connectivity

NAT A has assigned its UDP port 62000 to the communication session between A and S, and NAT B has assigned its port 31000 to the session between B and S. By communicating with server S, A and B learn each other's public endpoints as observed by S. Client A now starts sending UDP messages to port 31001 at address 192.0.2.254 (note the port number increment), and client B simultaneously starts sending messages to port 62001 at address 192.0.2.1. If NATs A and B assign port numbers to new sessions sequentially, and if not much time has passed since the A-S and B-S sessions were initiated, then a working bidirectional communication channel between A and B should result. A's messages to B cause NAT A to open up a new session, to which NAT A will (hopefully) assign public port number 62001, because 62001 is next in sequence after the port number 62000 it previously assigned to the session between A and S. Similarly, B's messages to A will cause NAT B to open a new session, to which it will (hopefully) assign port number 31001. If both clients have correctly guessed the port numbers each NAT assigns to the new sessions, then a bidirectional UDP communication channel will have been established.

Clearly, there are many things that can cause this trick to fail. If the predicted port number at either NAT already happens to be in use by an unrelated session, then the NAT will skip over that port number and the connection attempt will fail. If either NAT sometimes or always chooses port numbers non-sequentially, then the trick will fail. If a different client behind NAT A (or B, respectively) opens up a new outgoing UDP connection to any external destination after A (B) establishes its connection with S but before sending its first message to B (A), then the unrelated client will inadvertently "steal" the desired port number. This trick is therefore much less likely to work when either NAT involved is under load.

Since in practice an application implementing this trick would still need to work even when one of the NATs employs Endpoint-Independent Mapping, the application would need to detect beforehand what kind of NAT is involved on either end and modify its behavior accordingly, increasing the complexity of the algorithm and the general brittleness of the network. Finally, port number prediction has little chance of working if either client is behind two or more levels of NAT and the NAT(s) closest to the client employs Endpoint-Dependent Mapping.

## 3.6. TCP Port Number Prediction

This is a variant of the "TCP Hole Punching" technique to set up direct peer-to-peer TCP sessions across NATs employing Address-Dependent Mapping.

Unfortunately, this trick may be even more fragile and timing-sensitive than the UDP port number prediction trick described earlier. First, predicting the public port a NAT would assign could be wrong. In addition, if either client's SYN arrives at the opposite NAT device too quickly, then the remote NAT device may reject the SYN with a RST packet, causing the local NAT device in turn to close the new session and make future SYN retransmission attempts using the same port numbers futile.

## 4. Recent Work on NAT Traversal

[P2P-NAT] has a detailed discussion on the UDP and TCP hole punching techniques for NAT traversal. [P2P-NAT] also lists empirical results from running a test program [NAT-CHECK] across a number of commercial NAT devices. The results indicate that UDP hole punching works widely on more than 80% of the NAT devices, whereas TCP hole punching works on just over 60% of the NAT devices tested. The results also indicate that TCP or UDP hairpinning is not yet widely available on commercial NAT devices, as less than 25% of the devices passed the tests ([NAT-CHECK]) for Hairpinning. Readers may also refer to [JENN-RESULT] and [SAIK-RESULT] for empirical test results in classifying publicly available NAT devices. [JENN-RESULT] provides results of NAT classification using tests spanning across different IP protocols. [SAIK-RESULT] focuses exclusively on classifying NAT devices by the TCP behavioral characteristics.

[TCP-CHARACT] and [NAT-BLASTER] focus on TCP hole punching, exploring and comparing several alternative approaches. [NAT-BLASTER] takes an analytical approach, analyzing different cases of observed NAT behavior and ways applications might address them. [TCP-CHARACT] adopts a more empirical approach, measuring the commonality of different types of NAT behavior relevant to TCP hole punching. This work finds that using more sophisticated techniques than those used in [P2P-NAT], up to 88% of currently deployed NATs can support TCP hole punching.

[TEREDO] is a NAT traversal service that uses relay technology to connect IPv4 nodes behind NAT devices to IPv6 nodes, external to the NAT devices. [TEREDO] provides for peer communication across NAT devices by tunneling packets over UDP, across the NAT device(s) to a relay node. Teredo relays act as Rendezvous servers to relay traffic from private IPv4 nodes to the nodes in the external realm and vice versa.

[ICE] is a NAT traversal protocol for setting up media sessions between peer nodes for a class of multi-media applications. [ICE] requires peering nodes to run the Simple Traversal of the UDP Protocol through NAT (STUN) protocol [STUN] on the same port number used to terminate media session(s). Applications that use signaling protocols such as SIP ([SIP]) may embed the NAT traversal attributes for the media session within the signaling sessions and use the offer/answer type of exchange between peer nodes to set up end-to-end media session(s) across NAT devices. [ICE-TCP] is an extension of ICE for TCP-based media sessions.

A number of online gaming and media-over-IP applications, including Instant Messaging applications, use the techniques described in the document for peer-to-peer connection establishment. Some applications may use multiple distinct rendezvous servers for registration, discovery, and relay functions for load balancing, among other reasons. For example, the well-known media-over-IP application "Skype" uses a central public server for login and different public servers for end-to-end relay function.

# 5. Summary of Observations

## 5.1. TCP/UDP Hole Punching

TCP/UDP hole punching appears to be the most efficient existing method of establishing direct TCP/UDP peer-to-peer communication between two nodes that are both behind NATs. This technique has been used with a wide variety of existing NATs. However, applications may need to prepare to fall back to simple relaying when direct communication cannot be established.

The TCP/UDP hole punching technique has a caveat in that it works only when the traversing NAT is EIM-NAT. When the NAT device enroute is not EIM-NAT, the application is unable to reuse an already established endpoint mapping for communication with different external destinations and the technique would fail. However, many of the NAT devices deployed in the Internet are EIM-NAT devices. That makes the TCP/UDP hole punching technique broadly applicable [P2P-NAT]. Nevertheless, a substantial fraction of deployed NATs do employ Endpoint-Dependent Mapping and do not support the TCP/UDP hole punching technique.

## 5.2. NATs Employing Endpoint-Dependent Mapping

NATs Employing Endpoint-Dependent Mapping weren’t a problem with client-server applications such as Web browsers, which only need to initiate outgoing connections. However, in recent times, P2P applications such as Instant Messaging and Voice-over-IP have been in wide use. NATs employing Endpoint-Dependent Mapping are not suitable for P2P applications as techniques such as TCP/UDP hole punching will not work across these NAT devices.

### 5.3. Peer Discovery

Application peers may be present within the same NAT domain or external to the NAT domain. In order for all peers (those within or external to the NAT domain) to discover the application endpoint, an application may choose to register its private endpoints in addition to public endpoints with the rendezvous server.

### 5.4. Hairpinning

Support for hairpinning is highly beneficial to allow hosts behind EIM-NAT to communicate with other hosts behind the same NAT device through their public, possibly translated, endpoints. Support for hairpinning is particularly useful in the case of large-capacity NATs deployed as the first level of a multi-level NAT scenario. As described in Section 3.3.3, hosts behind the same first-level NAT but different second-level NATs do not have a way to communicate with each other using TCP/UDP hole punching techniques, unless the first-level NAT also supports hairpinning. This would be the case even when all NAT devices in a deployment preserve endpoint identities.

## 6. Security Considerations

This document does not inherently create new security issues. Nevertheless, security risks may be present in the techniques described. This section describes security risks the applications could inadvertently create in attempting to support direct communication across NAT devices.

### 6.1. Lack of Authentication Can Cause Connection Hijacking

Applications must use appropriate authentication mechanisms to protect their connections from accidental confusion with other connections as well as from malicious connection hijacking or denial-of-service attacks. Applications effectively must interact with multiple distinct IP address domains, but are not generally aware of the exact topology or administrative policies defining these address domains. While attempting to establish connections via TCP/UDP hole punching, applications send packets that may frequently arrive at an entirely different host than the intended one.

For example, many consumer-level NAT devices provide Dynamic Host Configuration Protocol (DHCP) services that are configured by default to hand out site-local IP addresses in a particular address range. Say, a particular consumer NAT device, by default, hands out IP addresses starting with 192.168.1.100. Most private home networks using that NAT device will have a host with that IP address, and many of these networks will probably have a host at address 192.168.1.101

as well. If host A at address 192.168.1.101 on one private network attempts to establish a connection by UDP hole punching with host B at 192.168.1.100 on a different private network, then as part of this process host A will send discovery packets to address 192.168.1.100 on its local network, and host B will send discovery packets to address 192.168.1.101 on its network. Clearly, these discovery packets will not reach the intended machine since the two hosts are on different private networks, but they are very likely to reach SOME machine on these respective networks at the standard UDP port numbers used by this application, potentially causing confusion, especially if the application is also running on those other machines and does not properly authenticate its messages.

This risk due to aliasing is therefore present even without a malicious attacker. If one endpoint, say, host A, is actually malicious, then without proper authentication the attacker could cause host B to connect and interact in unintended ways with another host on its private network having the same IP address as the attacker's (purported) private address. Since the two endpoint hosts A and B presumably discovered each other through a public rendezvous server S, providing registration, discovery, and limited relay services, and neither S nor B has any means to verify A's reported private address, applications may be advised to assume that any IP address they find to be suspect until they successfully establish authenticated two-way communication.

## 6.2. Denial-of-Service Attacks

Applications and the public servers that support them must protect themselves against denial-of-service attacks, and ensure that they cannot be used by an attacker to mount denial-of-service attacks against other targets. To protect themselves, applications and servers must avoid taking any action requiring significant local processing or storage resources until authenticated two-way communication is established. To avoid being used as a tool for denial-of-service attacks, applications and servers must minimize the amount and rate of traffic they send to any newly discovered IP address until after authenticated two-way communication is established with the intended target.

For example, applications that register with a public rendezvous server can claim to have any private IP address, or perhaps multiple IP addresses. A well-connected host or group of hosts that can collectively attract a substantial volume of connection attempts (e.g., by offering to serve popular content) could mount a denial-of-service attack on a target host C simply by including C's IP address in its own list of IP addresses it registers with the rendezvous server. There is no way the rendezvous server can verify the IP addresses, since they could well be legitimate private network addresses useful to other hosts for establishing network-local communication. The application protocol must therefore be designed to size- and rate-limit traffic to unverified IP addresses in order to avoid the potential damage such a concentration effect could cause.

## 6.3. Man-in-the-Middle Attacks

Any network device on the path between a client and a public rendezvous server can mount a variety of man-in-the-middle attacks by pretending to be a NAT. For example, suppose host A attempts to register with rendezvous server S, but a network-snooping attacker is able to observe this registration request. The attacker could then flood server S with requests that are identical to the client's original request except with a modified source IP address, such as the IP address of the attacker itself. If the attacker can convince the server to register the client using the attacker's IP address, then the attacker can make itself an active component on the path of all future traffic from the server AND other hosts to the original client, even if the attacker was originally only able to snoop the path from the client to the server.

The client cannot protect itself from this attack by authenticating its source IP address to the rendezvous server, because in order to be NAT-friendly the application must allow intervening NATs to change the source address silently. This appears to be an inherent security weakness of the NAT paradigm. The only defense against such an attack is for the client to authenticate and potentially encrypt the actual content of its communication using appropriate higher-level identities, so that the interposed attacker is not able to take advantage of its position. Even if all application-level communication is authenticated and encrypted, however, this attack could still be used as a traffic analysis tool for observing who the client is communicating with.

## 6.4. Security Impact from EIM-NAT Devices

Designing NAT devices to preserve endpoint identities does not weaken the security provided by the NAT device. For example, a NAT device employing Endpoint-Independent Mapping and Endpoint-Dependent Filtering is no more "promiscuous" than a NAT device employing Endpoint-Dependent Mapping and Endpoint-Dependent Filtering. Filtering incoming traffic aggressively using Endpoint-Dependent Filtering while employing Endpoint-Independent Mapping allows a NAT device to be friendly to applications without compromising the principle of rejecting unsolicited incoming traffic.

Endpoint-Independent Mapping could arguably increase the predictability of traffic emerging from the NAT device, by revealing the relationships between different TCP/UDP sessions and hence about the behavior of applications running within the enclave. This predictability could conceivably be useful to an attacker in exploiting other network- or application-level vulnerabilities. If the security requirements of a particular deployment scenario are so critical that such subtle information channels are of concern, then perhaps the NAT device was not to have been configured to allow unrestricted outgoing TCP/UDP traffic in the first place. A NAT device configured to allow communication originating from specific applications at specific ports, or via tightly controlled application-level gateways, may accomplish the security requirements of such deployment scenarios.

## 7. Acknowledgments

The authors wish to thank Henrik Bergstrom, David Anderson, Christian Huitema, Dan Wing, Eric Rescorla, and other BEHAVE work group members for their valuable feedback on early versions of this document. The authors also wish to thank Francois Audet, Kaushik Biswas, Spencer Dawkins, Bruce Lowekamp, and Brian Stucker for agreeing to be technical reviewers for this document.

## 8. References

### 8.1. Normative References

[NAT-TERM] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[NAT-TRAD] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[BEH-UDP] Audet, F., Ed., and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.

### 8.2. Informative References

[BEH-APP] Ford, B., Srisuresh, P., and D. Kegel, "Application Design Guidelines for Traversal through Network Address Translators", Work in Progress, March 2007.

[BEH-ICMP] Srisuresh, P., Ford, B., Sivakumar, S., and S. Guha, "NAT Behavioral Requirements for ICMP protocol", Work in Progress, February 2008.

[BEH-TCP] Guha, S., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", Work in Progress, April 2007.

[BIDIR] Peer-to-Peer Working Group, NAT/Firewall Working Committee, "Bidirectional Peer-to-Peer Communication with Interposing Firewalls and NATs", August 2001. http://www.peer-to-peerwg.org/tech/nat/

[ICE] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", Work in Progress, October 2007.

[ICE-TCP] Rosenberg, J., "TCP Candidates with Interactive Connectivity Establishment (ICE)", Work in Progress, July 2007.

[JENN-RESULT] Jennings, C., "NAT Classification Test Results", Work in Progress, July 2007.

[KEGEL] Kegel, D., "NAT and Peer-to-Peer Networking", July 1999. http://www.alumni.caltech.edu/~dank/peer-nat.html

[MIDCOM] Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A., and A. Rayhan, "Middlebox communication architecture and framework", RFC 3303, August 2002.

[NAT-APPL] Senie, D., "Network Address Translator (NAT)-Friendly Application Design Guidelines", RFC 3235, January 2002.

[NAT-BLASTER] Biggadike, A., Ferullo, D., Wilson, G., and Perrig, A., "Establishing TCP Connections Between Hosts Behind NATs", ACM SIGCOMM ASIA Workshop, April 2005.

[NAT-CHECK] Ford, B., "NAT check Program" available online as http://midcom-p2p.sourceforge.net, February 2005.

[NAT-PMP] Cheshire, S., Krochmal, M., and K. Sekar, "NAT Port Mapping Protocol (NAT-PMP)", Work in Progress, October 2006.

[NAT-PROT] Holdrege, M. and P. Srisuresh, "Protocol Complications with the IP Network Address Translator", RFC 3027, January 2001.

[NAT-PT] Tsirtsis, G. and P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)", RFC 2766, February 2000.

[NAT-PT-HIST] Aoun, C. and E. Davies, "Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status", RFC 4966, July 2007.

[NSIS-NSLP] Stiemerling, M., Tschofenig, H., Aoun, C., and E. Davies, "NAT/Firewall NSIS Signaling Layer Protocol (NSLP)", Work in Progress, July 2007.

[P2P-NAT] Ford, B., Srisuresh, P., and Kegel, D., "Peer-to-Peer Communication Across Network Address Translators", Proceedings of the USENIX Annual Technical Conference (Anaheim, CA), April 2005.

[RFC3330] IANA, "Special-Use IPv4 Addresses", RFC 3330, September 2002.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.

[RSIP] Borella, M., Lo, J., Grabelsky, D., and G. Montenegro, "Realm Specific IP: Framework", RFC 3102, October 2001.

[SAIK-RESULT] Guha, Saikat, "NAT STUNT Results" available online as https://www.guha.cc/saikat/stunt-results.php.

[SIP] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[SOCKS] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and L. Jones, "SOCKS Protocol Version 5", RFC 1928, March 1996.

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

[SYM-STUN] Takeda, Y., "Symmetric NAT Traversal using STUN", Work in Progress, June 2003.

[TCP] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[TCP-CHARACT] Guha, S., and Francis, P., "Characterization and Measurement of TCP Traversal through NATs and Firewalls", Proceedings of Internet Measurement Conference (IMC), Berkeley, CA, October 2005, pp. 199-211.

[TEREDO] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, February 2006.

[TURN] Rosenberg, J., Mahy, R., and P. Matthews, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", Work in Progress, January 2008.

[UNSAF] Daigle, L., Ed., and IAB, "IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation", RFC 3424, November 2002.

[UPNP] UPnP Forum, "Internet Gateway Device (IGD) Standardized Device Control Protocol V 1.0", November 2001, http://www.upnp.org/standardizeddcps/igd.asp

[V6-CPE-SEC] Woodyatt, J., "Recommended Simple Security Capabilities in Customer Premises Equipment for Providing Residential IPv6 Internet Service", Work in Progress, June 2007.

## Authors' Addresses

Pyda Srisuresh
Kazeon Systems, Inc.
1161 San Antonio Rd.
Mountain View, CA 94043
USA

Phone: (408)836-4773
EMail: srisuresh@yahoo.com

Bryan Ford
Laboratory for Computer Science
Massachusetts Institute of Technology
77 Massachusetts Ave.
Cambridge, MA 02139
USA

Phone: (617) 253-5261
EMail: baford@mit.edu
Web: http://www.brynosaurus.com/

Dan Kegel
Kegel.com
901 S. Sycamore Ave.
Los Angeles, CA 90036
USA

Phone: 323 931-6717
EMail: dank@kegel.com
Web: http://www.kegel.com/