

# NAT Traversal Techniques in Peer-to-Peer Networks

Huynh Cong Phuoc  
University of Canterbury  
Christchurch, New Zealand  
huynh@rsise.anu.edu.au

Ray Hunt  
University of Canterbury  
Christchurch, New Zealand  
ray.hunt@canterbury.ac.nz

Andrew McKenzie  
University of Canterbury  
Christchurch, New Zealand  
apm60@student.canterbury.ac.nz

## ABSTRACT

Peer to peer (P2P) networking has important applications in Internet telephony, online gaming, multimedia communication, instant messaging, file and workspace sharing which raises the need for robust and reliable NAT traversal techniques. This paper discusses current and developing techniques and challenges posed by NAT traversal in P2P networks. Initially Network Address Translation (NAT) detection is categorised and both UDP and TCP traversal techniques are discussed. Methodologies such as Relaying, Connection Reversal, and Hole Punching are then analysed. Finally the development of a testbed is described which can be used to evaluate NAT traversal techniques and to determine appropriate configurations in order to achieve P2P networking.

## Categories and Subject Descriptors

C.2.6 [Computer Systems Organization]: Communication/Networking and Information Technology/Internet networking

## General Terms

NAT Traversal

## Keywords

NAT Traversal, Peer to Peer Networks

## 1. INTRODUCTION

Recent years have seen a significant rise in the use and development of Peer to Peer networks. Protocols such as Bit Torrent and Gnutella allow for the distributed sharing of files, while VOIP applications such as Skype(tm) provide the ability to communicate with clarity comparable to that of a landline.

A major obstacle encountered by P2P clients is the presence of NAT which allows the mapping of a large pool of internal addresses to a limited pool of external addresses. Inconsistencies, however, arise in the way NAT handles unsolicited

This paper was published in the proceedings of the New Zealand Computer Science Research Student Conference 2008. Copyright is held by the author/owner(s).

NZCSRSC 2008, April 2008, Christchurch, New Zealand.

inbound connections. NAT traversal techniques are therefore needed by P2P applications to establish connections between clients obstructed by NAT. This paper investigates a number of these techniques.

In order to determine the most appropriate means of establishing a connection, obstructing NATs are classified by the forms of inbound connections they accept. STUN[7] and STUNT[3] provide this detection and classification capability for UDP and TCP respectively. Section 2 of this paper outlines the various classifications they provide, and Section 4 provides an overview of a testbed trial created to demonstrate the capabilities of STUN.

Various techniques exist for the traversal of NAT by P2P applications. TCP and UDP Hole Punching use a method of simultaneously attempting to establish connections between clients where both are obstructed by NAT. This is done after first attempting to establish a connection on the private network, creating a more efficient connection and avoiding the problem of hairpinning. STUN is presented as a way to detect the presence of NAT, and the feasibility of establishing a direct public connection. Finally, TURN[8] presents a fallback method of relaying packets through an intermediary node. These techniques are discussed in detail in Section 3.

## 2. NAT DETECTION AND CATEGORISATION

RFC3022 [9] summarises (Traditional) NAT as a system that would allow hosts within a private network to transparently access hosts in the external network. Essentially, NAT allows a small pool of public IP addresses to be used by a large number of clients. This divides a network into two realms: a public realm where a client uses an IP address assigned by the NAT device; and a private realm where typically either IP addresses within the private address space, or increasingly, IPv6 addresses are used.

RFC3022 also draws the distinction between Basic NAT and Network Address/Port Translation (NAPT). Basic NAT allows for one-to-one address mapping, i.e. if a node in the private realm wishes to access the public realm then the NAT device assigns it a public IP address for that session. NAPT, on the other hand, maps between public and private IP:Port combinations. This allows for the efficient use of available public IP addresses, as many such addresses may be mapped to the one public address.

As mappings are generally established in an ad-hoc manner,

NAT's primary functions focus on outbound connections. Whilst port forwarding is usually possible for incoming connections, generally this requires a level of administrative access outside the reach of a client on a large network. Thus determining the types of NAT will help evaluating the applicability of each NAT traversal technique for P2P communications.

The classification of NATs that is relevant to P2P communications is based on their address (and port) binding and incoming packet filtering mechanisms. This categorisation depends on the type of transport protocol (UDP or TCP) carrying the P2P packets and the following categorisations of NAT behaviour with UDP and TCP traffic are described.

### 2.1 NAT categorisation for UDP traffic

There are four types of NAT depending on their address binding schemes for UDP traffic, as identified in RFC3489

- Full Cone. A full cone NAT maps an internal IP address and port to the same public external IP address and port for any outgoing packet regardless of the destination address and port. In addition, any external host can send packets to the internal host through the mapped external address.
- Restricted Cone. A restricted cone NAT also maps an internal IP address and port to the same public external IP address and port. However, an external host (with address X) can only send a packet to an internal host through this mapped address and port if the internal host previously sent a packet to the IP address X.
- Port Restricted Cone. A port restricted cone NAT further filters incoming packets. Specifically, it only accepts incoming packets from an external host with address X and port P if an internal host has specifically sent a packet to this address:port combination.
- Symmetric. A symmetric NAT creates a different address mapping of the same internal address and port for each destination IP address and port. Thus an external host can only send a UDP packet back to the internal host through the mapped public address and port that was created for a previous transmission from the internal host to the external host.

### 2.2 NAT categorisation for TCP traffic

The following categories of NAT devices for TCP traffic have been described in a draft version on STUNT.

- Independent. The NAT device maps the same internal transport address (IP address and port) to the same external transport address for all outbound TCP connections originating from that internal transport address regardless of their destination.
- Address Dependent. The NAT device maps all outbound TCP connections from the same internal source transport address to the same external source transport addresses only when the destination IP addresses are the same.

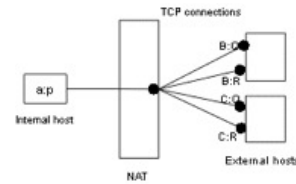


Figure 1: Independent NAT

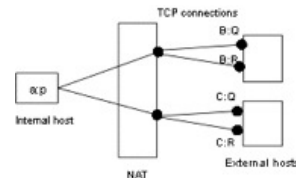


Figure 2: Address Dependent NAT

- Address and Port Dependent. The NAT device maps all outbound TCP connections from the same internal source transport address to the same external source transport addresses only when the destination IP addresses and ports are the same.
- Session Dependent. The NAT maps each new outbound TCP connection from the same private source transport address to a different external source transport address even if the destination IP addresses and ports are the same.

## 3. NAT TRAVERSAL TECHNIQUES

### 3.1 NAT Traversal and UDP Hole Punching

The NAT traversal mechanism of UDP packets consists of two-phases. In the first phase, each peer behind a NAT discovers the presence and types of NATs and firewalls between them and the Internet. This discovery includes the NATs' treatment of UDP traffic and the public IP address and port assigned to the peer. In the second phase, the public address and the NAT behaviour obtained in the first phase are used to predict the address and port number for a subsequent session between the peers. This technique is known as hole punching [2].

Hole punching is a NAT traversal technique used to establish direct P2P communications between hosts which does not compromise the security of a private network. Furthermore, it respects the default security policy of most NATs. In fact, UDP hole punching does not establish any unsolicited sessions between peers. This technique has been implemented in experimental protocols such as Teredo [5] and ICE [6]. A

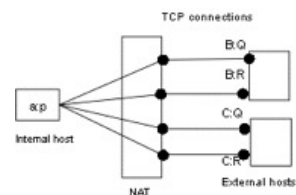


Figure 3: Address and Port Dependent NAT

number of proprietary protocols, such as those for on-line gaming, also use UDP hole punching.

The prerequisite of UDP hole punching is that the two peers have had active UDP sessions with a common rendezvous server which records the private and public IP address and port of each peer. The private address and port are stored in a field of the peer's registration message, and the public address and port are the source IP address and port fields in the UDP header of this message.

Hairpinning is a problem which occurs when two clients on the same private network attempt to establish a connection via a NAT device on the public network. Hole punching remedies this by having the server send the private network addresses to the respective clients. The clients initially attempt to establish a connection using these private network addresses thus avoiding the problem of hairpinning. To establish a UDP session between two peers A and B through a rendezvous server S, the following process occurs:

1. A asks S for help establishing a UDP session with B.
2. S replies to A with the public and private addresses of B. At the same time, S uses its UDP session with B to forward to B the connection request containing A's public and private addresses. At the end of this step, both the peers know the addresses of each other.
3. When a peer (A or B) receives the public and private addresses of the other, it starts sending UDP packets to both of these addresses and becomes connected to whichever address that responds to it first. The initial UDP packets are sent asynchronously, and the order and timing between them are not important.

The remaining steps of the UDP session establishment process vary in each specific network scenario. Three specific network scenarios are common: (i) Peers are behind a common NAT (ii) Peers are behind different NATs and (iii) Peers are behind multiple levels of NATs and share a common NAT.

### 3.1.1 NAT Type Detection with STUN (Simple Traversal of UDP Through NATs)

STUN is a client-server protocol, outlined in RFC3489 and provides a means of detecting which type of NAT (if any) a machine is concealed by as well as its external address information. By establishing a connection with a STUN server either over a direct or secure link, a client is able to request information (via a binding request) pertaining to its NAT status. The STUN server then proceeds to probe the client via the external address used to connect with the server. This process is demonstrated in more detail in the description of the testbed (Section 4).

To probe the client, two external sources are used: the primary STUN server, and a secondary server using a different IP address. Dependant on the request sent by the client, each server proceeds to transmit UDP packets to the external address from which it was sourced. Packets from the server are also transmitted from differing port numbers to differentiate between restricted cone and port restricted cone

NAT. The client also has the option of connecting to the second external source to determine if it is behind a symmetric NAT.

By the use of STUN, a client is therefore able to:

- Determine its external IP address
- Determine what type of NAT it is behind
- Achieve this with limited user knowledge or input

### 3.1.2 TURN (Traversal Using Relay NAT)

TURN (currently available as a draft version) is an extension of STUN to overcome STUN's limitations in situations where addresses returned by the STUN server are not usable by peers behind symmetric NATs or the peers are behind the same NAT. The TURN protocol introduces a relay usage of STUN, that allocates to each client an IP address on the server's interface on which packets from its peers are received. The server is located on the public side of the NAT and acts as a relay server. When it receives a packet on the allocated address, it forwards the packet to the client. This method guarantees to work even when the NATs are symmetric because the P2P traffic just travels along the old (relay) route through the server. Therefore there is no need for a new UDP session through the NATs.

## 3.2 NAT Traversal with TCP

P2P communications over TCP connections are more robust than they are over UDP for the following reasons. UDP is unreliable because it is connectionless and has no explicit end-to-end communications. In addition, NATs can timeout a UDP port mapping after a period of inactivity. To maintain the address mapping, keep-alive messages must be sent to NAT periodically. Unlike UDP, the TCP state machine gives NATs on the path a way to determine the precise lifetime of a particular TCP session. P2P communications over TCP uses a reliable connection with error detection and packet retransmission.

The mechanism of NAT traversal of TCP traffic is more complicated than that of UDP traffic. Each host behind a NAT has to not only detect the presence of NATs and predict the public address and port assigned to it by the NATs, but also obtain the initial sequence number in the first SYN packet sent by the other host to establish a TCP connection. Therefore, TCP traversal techniques have to manage the synchronisation of the sequence numbers of TCP packets sent from both peers.

### 3.2.1 TCP hole punching

TCP hole punching operates in a similar fashion to UDP hole punching, with a slight complexity to establish the TCP handshake process between the peers. As with UDP, TCP hole punching also assumes that each peer (A and B) has an active TCP connection with the same rendezvous server S. In addition, the peers register their private and public addresses on the server.

In general, TCP hole punching (3.2.1) proceeds as follows.

- A uses its connection with S to ask S for a connection with B.

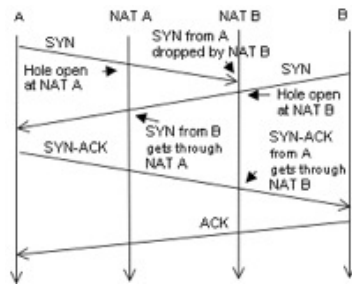


Figure 4: TCP Hole Punching operation

- S replies to A with B's private and public addresses, and simultaneously sends A's addresses to B.
- A and B asynchronously make outgoing connection attempts (send SYN packets) to each other's public and private addresses, from the same port that they used to register with S. At the same time, they listen for TCP incoming connection attempts on their local TCP ports.
- A and B wait for a SYN-ACK response to their outgoing SYN packets, or an incoming connection request (SYN packet). If a connection fails, the peer can retry it up to a maximum timeout period.
- Once the three-way handshake process has completed, the peers authenticate each other. If the authentication fails, the peers close that connection and wait until another connection is successfully authenticated. The first successfully authenticated connection will be used to transfer TCP data.

In each network scenario, TCP hole punching operates in a similar way to UDP hole punching. For example, if two peers A and B are behind different NATs, each peer's first SYN packet sent to the other peer opens up a hole associated with its public address in its respective NAT. If A's first SYN packet to B reaches B's NAT before B's first SYN packet to A reaches B's NAT, B's NAT considers A's SYN packet unsolicited and drops it. However, subsequently B's first SYN packet can travel through A's NAT successfully because A's NAT recognises B's public address as the destination of the outgoing session that A has initiated.

Recent implementations of this technique in different protocols and systems include Simple Traversal of UDP Through NATs and TCP too (STUNT), NUTSS [4] and NATBlaster [1].

#### 4. FURTHER RESEARCH

We intend to implement an as yet unpublished protocol called PNT (Peer to Peer NAT Traversal). This protocol, authored by Pyda Srisuresh, combines a number of the techniques described in this paper to form a general framework able to be used by most P2P applications. The implementation of this framework is still in its early stages, and is expected to be completed mid 2008.

#### 5. CONCLUSIONS

The traversal of UDP and TCP traffic through NATs poses a variety of challenges and difficulties. Lack of standardisation of NATs makes it impossible to find a comprehensive solution that applies to all NAT behaviours in all networking scenarios. Specifically, NATs differ in their address and port binding schemes and their treatment of incoming traffic. Solving the problem for TCP traffic appears to be more difficult due to the additional processes required to establish the TCP handshake, and to synchronise the packet sequence numbers.

This paper has discussed a number of NAT traversal techniques but solutions vary due to the variations in NAT implementation and the characteristics of network scenarios. Limitations also occur as a result of security threats, the variety NAT behaviours, excessive network resource requirements (TURN) and the computational complexity of the some implementations (NATBlaster). Currently there are only limited solutions in these situations and further research is necessary, particularly in the areas of random port and address allocation and limitations resulting from lack of hairpin translation.

#### 6. REFERENCES

- [1] G. W. A. Biggadike, D. Ferullo, and A. Perrig. Natblaster: Establishing tcp connections between hosts behind nats. *Proceedings of ACM SIGCOMM Asia Workshop*, April 2005.
- [2] B. Ford, P. Srisuresh, , and D. Kegel. Peer-to-peer communication across network address translators. *Proceedings of USENIX Annual Technical Conference*, April 2005.
- [3] S. Guha. Internet-draft: Stunt - simple traversal of udp through nats and tcp too. *Technical report, Cornell University*, December 2004.
- [4] S. Guha, Y. Takeda, and P. Francis. Nutss: A sip-based approach to udp and tcp network connectivity. *Proceedings of SIGCOMM 2004 Workshops*, August 2004.
- [5] C. Huitema. Rfc 4380: Teredo: Tunneling ipv6 over udp through network address translations (nats). *Technical report*, February 2006.
- [6] J. Rosenberg. Interactive connectivity establishment (ice): A methodology for network address translator (nat) traversal for offer/answer protocols. *Technical report, The Internet Society*, March 2006.
- [7] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. Rfc 3489: Stun - simple traversal of user datagram protocol (udp) through network address translators (nats). *Technical report, The Internet Society*, March 2003.
- [8] J. Rosenberg, R. Mahy, and C. Huitema. Obtaining relay addresses from simple traversal of udp through nat (stun). *Technical report, The Internet Society*, February 2006.
- [9] P. Srisuresh and K. Egevang. Traditional ip network address translator (traditional nat). *RFC 3022, Internet Engineering Task Force*, January 2001.