SIPPING Working Group C. Boulton
Internet-Draft Ubiquity Software Corporation
Expires: April 4, 2005 J. Rosenberg
Cisco Systems
October 2004

Best Current Practices for NAT Traversal for SIP
draft-ietf-sipping-nat-scenarios-02

Status of this Memo

Copyright Notice

Abstract

Traversal of the Session Initiation Protocol (SIP) and the sessions it establishes through Network Address Translators (NAT) is a complex problem. Currently there are many deployment scenarios and traversal mechanisms for media traffic. This document aims to provide concrete recommendations and a unified method for NAT traversal as well as

documenting corresponding call flows.

Table of Contents

1. Introduction 3
2. Problem Statement 3
3. Solution Technology Outline Description 6
3.1 SIP Signaling 7
3.1.1 Symmetric Response 7
3.1.2 Connection Re-use 8
3.2 Media Traversal 8
3.2.1 Symmetric RTP 8
3.2.2 STUN 9
3.2.3 TURN 9
3.2.4 ICE 9
3.2.5 RTCP Attribute 10
3.2.6 Solution Profiles 10
4. NAT Traversal Scenarios 11
4.1 Basic NAT SIP Signaling Traversal 11
4.1.1 Registration (Registrar/Proxy Co-Located 11
4.1.2 Registration(Registrar/Proxy not Co-Located) 15
4.1.3 Initiating a Session 16
4.1.4 Receiving an Invitation to a Session 18
4.2 Basic NAT Media Traversal 21
4.2.1 Port Restricted Cone NAT 21
4.2.2 Symmetric NAT 33
4.3 Advanced NAT media Traversal Using ICE 36
4.3.1 Full Cone --> Full Cone traversal 37
4.3.2 Port Restricted Cone --> Port Restricted Cone traversal 37
4.3.3 Internal TURN Server (Enterprise Deployment) 37
4.4 Intercepting Intermediary (B2BUA) 37
4.5 IPV4/IPV6 37
4.6 ICE with RTP/TCP 37
5. References 37
5.1 Normative References 37
5.2 Informative References 39
Authors' Addresses 39
Intellectual Property and Copyright Statements 40

1. Introduction

NAT (Network Address Translators) traversal has long been identified as a large problem when considered in the context of the Session Initiation Protocol (SIP)[1] and it's associated media such as Real Time Protocol (RTP)[2]. The problem is further confused by the variety of NATs that are available in the market place today and the large number of potential deployment scenarios. Detail of different NAT types can be found in RFC 3489bis [14].

The IETF has produced many specifications for the traversal of NAT, including STUN, ICE, rport, symmetric RTP, TURN, connection reuse, SDP attribute for RTCP, and others. These each represent a part of the solution, but none of them gives the overall context for how the NAT traversal problem is decomposed and solved through this collection of specifications. This document serves to meet that need.

This document attempts to provide a definitive set of 'Best Common Practices' to demonstrate the traversal of SIP and it's associated media through NAT devices. The document does not propose any new functionality but does draw on existing solutions for both core SIP signaling and media traversal (as defined in section 3).

The draft will be split into distinct sections as follows:

1. A clear definition of the problem statement
2. Description of proposed solutions for both SIP protocol signaling and media signaling
3. A set of basic and advanced call flow scenarios

2. Problem Statement

The traversal of SIP through NAT can be split into two categories that both require attention - The core SIP signaling and associated media traversal.

The core SIP signaling has a number of issues when traversing through NATs.

Firstly, the default operation for SIP response generation using unreliable protocols such as the Unicast Datagram Protocol (UDP) results in responses being generated at the User Agent Server (UAS) being sent to the source address, as specified in either the SIP 'Via' header or the 'received' parameter (as defined in RFC 3261 [1]). The port is extracted from the SIP 'Via' header to complete the IP address/port combination for returning the SIP response. While the destination is correct, the port contained in the SIP 'Via' header represents the listening port of the originating client and

not the port representing the open pin hole on the NAT. This results in responses being sent back to the NAT but to a port that is likely not open for SIP traffic. The SIP response will then be dropped at the NAT. This is illustrated in Figure 1 which depicts a SIP response being returned to port 5060.

```
    Private Network                   NAT                      Public Network
                                       |
                                       |
                                       |
     --------     SIP Request          |open port 5650            --------
    |        |----------------------->--->-----------------------|        |
    |        |                         |                         |        |
    | Client |                         |port 5060   SIP Response | Proxy  |
    |        |                         x<------------------------|        |
    |        |                         |                         |        |
     --------                          |                          --------
                                       |
                                       |
                                       |
```

Figure 1

Secondly, when using a reliable, connection orientated transport protocol such as TCP, SIP has an inherent mechanism that results in SIP responses reusing the connection that was created/used for the corresponding transactional request. The SIP protocol does not provide a mechanism that allows new requests generated in the opposite direction (Previously occupying the role of UAS for the last transaction) to use the existing TCP connection created between the client and the server during registration. This results in the registered contact address not being bound to the "connection" in the case of TCP. Requests are then blocked at the NAT, as illustrated in Figure 2. This problem also exists for unreliable transport protocols such as UDP where external NAT mappings need to be re-used to reach a SIP entity on the private side of the network.

```
    Private Network                   NAT                      Public Network
                                       |
                                       |
                                       |
     -------- (UAC 8023)       REGISTER/Response       (UAS 5060) --------
    |        |-----------------------&gt;---&lt;-----------------------|        |
    |        |                         |                         |        |
    | Client |                         |5060  INVITE   (UAC 8015)| Proxy  |
    |        |                         x&lt;------------------------|        |
    |        |                         |                         |        |
     --------                          |                          --------
                                       |
                                       |
                                       |
```

Figure 2

In Figure 2 the original REGISTER request is sent from the client on port 8023 and received on port 5060, establishing a reliable connection and opening a pin-hole in the NAT. The generation of a new request from the proxy results in a request destined for the registered entity (Contact IP address) which is not reachable from the public network. This results in the new SIP request attempting to create a connection to a private network address. This problem would be solved if the original connection was re-used. While this problem has been discussed in the context of connection orientated protocols such as TCP, the problem exists for SIP signaling using any transport protocol. The solution proposed for this problem in section 3 of this document is relevant for all SIP signaling, regardless of the transport protocol.

NAT policy can dictate that connections should be closed after a period of inactivity. This period of inactivity can range drastically from a number seconds to hours. Pure SIP signaling can not be relied upon to keep alive connections for a number of reasons. Firstly, SIP entities can sometimes have no signaling traffic for long periods of time which has the potential to exceed he inactivity timer, this can lead to problems where endpoints are not available to receive incoming requests as the connection has been closed. Secondly, if a low inactivity timer is specified, SIP signaling is not appropriate as a keep-alive mechanism as it has the potential to add a large amount of traffic to the network which uses up valuable resource and also requires processing at a SIP stack, which is also a waste of processing resource.

Media associated with SIP calls also has problems traversing NAT. RTP [2]] is on if the most common media transport type used in SIP

signaling. Negotiation of RTP occurs with a SIP session establishment using the Session Description Protocol(SDP) [3] and a SIP offer/answer exchange[4]. During a SIP offer/answer exchange an IP address and port combination are specified by each client in a session as a means of receiving media such as RTP. The problem arises when a client advertises it's address to receive media and it exists in a private network that is not accessible from outside the NAT. Figure 3 illustrates this problem.

```
              NAT             Public Network            NAT
               |                                         |
               |                                         |
               |                                         |
  --------     |            SIP Signaling Session        |    --------
 |        |-----------------------3>---<-----------------------|        |
 |        |    |                                         |    |        |
 | Client |    |                                         |    | Client |
 |   A    |>=========>RTP>=====================>RTP>======X    |   B    |
 |        |    X=====<RTP<=====================<RTP<=========<|        |
  --------     |                                         |     --------
               |                                         |
               |                                         |
               |                                         |
```

Figure 3

The connection address representing both clients are not available on the public internet and traffic can be sent from both clients through their NATs. The problem occurs when the traffic attempts to traverse media through the foreign (not local) NAT. The connection address extracted from the SDP payload is that of an internal address, and so not resolvable from the public side of the NAT. To complicate the problem further, a number of different NAT topologies with different default behaviors increase the difficulty of proposing a single solution.

3. Solution Technology Outline Description

When analyzing issues associated with traversal of SIP through existing NAT, it has been identified that the problem can be split into two clear solution areas as defined in section 2 of this document. The traversal of the core protocol signaling and the traversal of the associated media as specified in the Session Description Payload (SDP) of a SIP offer/answer exchange[4]. The following sub-sections outline solutions that enable core SIP signaling and its associated media to traverse NATs.

3.1 SIP Signaling

SIP signaling has two areas that result in transactional failure when traversing through NAT, as described in section 2 of this document. The remaining sub-sections describe appropriate solutions that result in SIP signalling traversal through NAT, regardless of transport protocol. IT is RECOMMEDED that SIP compliant entities follow the guidelines presented in this section to enable traversal of SIP signaling through NATs.

3.1.1 Symmetric Response

As described in section 2 of this document, when using an unreliable transport protocol such as UDP, SIP responses are sent to the IP address and port combination contained in the SIP 'Via' header field (or default port for the appropriate transport protocol if not present). This can result in responses being blocked at a NAT. In such circumstances, SIP signaling requires a mechanism that will allow entities to override the basic response generation mechanism in RFC 3261 [1]. Once the SIP response is constructed, the destination is still derived using the mechanisms described in RFC 3261 [1]. The port (to which the response will be sent), however, will not equal that specified in the SIP 'Via' header field but will be the port from which the original request was sent. This results in the pin-hole opened for the requests traversal of the NAT being reused, in a similar manner to that of reliable connection orientated transport protocols such as TCP. Figure 4 illustrates the response traversal through the open pin hole using this method.

```
    Private Network                    NAT                        Public Network
                                        |
                                        |
                                        |
     --------                           |                            --------
    |        |                          |                           |        |
    |        |send/receive              |               send/receive|        |
    | Client |port 5060-----<<->>-------------<<->>-----port 5060| Client |
    |   A    |                          |                           |   B    |
    |        |                          |                           |        |
     --------                           |                            --------
                                        |
                                        |
                                        |
```

Figure 4

The exact functionality for this method of response traversal is called 'Symmetric Response' and the details are documented in RFC 3581 [5]. Additional requirements are imposed on SIP entities in this specification such as listening and sending SIP requests/responses from the same port.

3.1.2 Connection Re-use

The second problem with sip signaling, as defined in Section 3.1.2, is to allow incoming requests to be properly routed. This is addressed in [9], which allows the reuse of a TCP connection or UDP 5-tuple for incoming requests. That draft also provides keepalive mechanisms based on using STUN to the SIP server. Usage of this specification is RECOMMENDED. This mechanism is not transport specific and should be used for any transport protocol.

Even if this draft is not used, clients SHOULD use the same IP address and port (i.e., socket) for both transmission and receipt of SIP messages. Doing so allows for the vast majority of industry provided solutions to properly function.

3.2 Media Traversal

This document has already provided guidelines that recommend using extensions to the core SIP protocol to enable traversal of NATs. While ultimately not desirable, the additions are relatively straight forward and provide a simple, universal solution for varying types of NAT deployment. The issues of media traversal through NATs is not as straight forward and requires the combination of a number of traversal methodologies. The technologies outlined in the remainder of this section provide the required solution set.

3.2.1 Symmetric RTP

The primary problem identified in section 2 of this document is that internal IP address/port combinations can not be reached from the public side of a NAT. In the case of media such as RTP, this will result in no audio traversing a NAT(as illustrated in Figure 3). To overcome this problem, a technique called 'Symmetric' RTP can be used. This involves an SIP endpoint both sending and receiving RTP traffic from the same IP Address/Port combination. This technique also requires intelligence by a client on the public internet as it identifies that incoming media for a particular session does not match the information that was conveyed in the SDP. In this case the client will ignore the SDP address/port combination and return RTP to the IP address/port combination identified as the source of the incoming media. This technique is known as 'Symmetric RTP' and is documented in [12]. 'Symmetric RTP' SHOULD only be used for

traversal of RTP through NAT when one of the participants in a media session definitively knows that it is on the public network.

3.2.2 STUN

Simple Traversal of User Datagram Protocol(UDP) through Network Address Translators(NAT) or STUN is defined in RFC 3489 [8]. It provides a lightweight protocol that allows entities to probe and discover the type of NAT that exist between itself and external entities. It also provides details of the external IP address/port combination used by the NAT device to represent the internal entity on the public facing side of a NAT. On learning of such an external representation, a client can use accordingly as the connection address in SDP to provide NAT traversal. STUN only works with Full Cone, Restricted Cone and Port Restricted Cone type NATs. STUN does not work with Symmetric NATs as the technique used to probe for the external IP address port representation using a STUN server will provide a different result to that required for traversal by an alternative SIP entity. The IP address/port combination deduced for the STUN server would be blocked for incoming packets from an alterative SIP entity.

3.2.3 TURN

As mentioned in the previous section, the STUN protocol does not work for UDP traversal through a Symmetric style NAT. Traversal Using Relay NAT (TURN) provides the solution for UDP traversal of symmetric NAT. TURN is extremely similar to STUN in both syntax and operation. It provides an external address at a TURN server that will act as a relay and guarantee traffic will reach the associated internal address. The full details of the TURN specification are defined in [11]. A TURN service will almost always provide media traffic to a SIP entity but it is RECOMMENDED that this method only be used as a last resort and not as a general mechanism for NAT traversal. This is because using TURN has high performance costs when relaying media traffic.

3.2.4 ICE

Interactive Connectivity Establishment (ICE) is the RECOMMENDED method for traversal of existing NAT if Symmetric RTP is not appropriate. ICE is a methodology for using existing technologies such as STUN, TURN and any other UNSAF[7] compliant protocol to provide a unified solution. This is achieved by obtaining as many representative IP address/port combinations as possible using technologies such as STUN/TURN etc. Once the addresses are accumulated, they are all included in the SDP exchange in a new media attribute called 'candidate'. Each 'candidate' SDP attribute entry

has detailed connection information including a media addresses (including optional RTCP information), priority, username, password and a unique session ID. The appropriate IP address/port combinations are used in the correct order depending on the specified priority. A client compliant to the ICE specification will then locally run instances of STUN servers on all addresses being advertised using ICE. Each instance will undertake connectivity checks to ensure that a client can successfully receive media on the advertised address. Only connections that pass the relevant connectivity checks are used for media exchange. The full details of the ICE methodology are contained in [13].

3.2.5 RTCP Attribute

Normal practice when selecting a port for defining Real Time Control Protocol(RTCP) [2] is for consecutive order numbering (i.e select an incremented port for RTCP from that used for RTP). This assumption causes RTCP traffic to break when traversing many NATs due to blocked ports. To combat this problem a specific address and port need to be specified in the SDP rather than relying on such assumptions. RFC 3605 [5] defines an SDP attribute that is included to explicitly specify transport connection information for RTCP. The address details can be obtained using any appropriate method including those detailed previously in this section (e.g. STUN, TURN).

3.2.6 Solution Profiles

This draft has documented a number of technology solutions for the traversal of media through differing NAT deployments. A number of 'profiles' will now be defined that categorize varying levels of support for the technologies described.

3.2.6.1 Primary Profile

A client falling into the 'Primary' profile supports ICE in conjunction with STUN, TURN and RFC 3605 [5] for RTCP. ICE is used in all cases and falls back to standard operation when dealing with non-ICE clients. A client which falls into the 'Primary' profile will be maximally interoperable and function in a rich variety of environments including enterprise, consumer and behind all variety of NAT.

3.2.6.2 Consumer Profile

A client falling into the 'Consumer' profile supports STUN and RFC 3605 [5] for RTCP. It uses STUN to allocate bindings, and can also detect when it is in the unfortunate situation of being behind a 'Symmetric' NAT, although it simply cannot function in this case.

These clients will only work in deployment situations where the access is sufficiently controlled to know definitively that there won't be Symmetric NAT. This is hard to guarantee as users can always pick up their client and connect via a different access network.

3.2.6.3 Minimal Profile

A client falling into the 'Minimal' profile will send/receive RTP form the same IP/port combination. This client requires proprietary network based solutions to function in any NAT traversal scenario.

All clients SHOULD support the 'Primary Profile', MUST support the 'Minimal Profile' and MAY support the 'Consumer Profile'.

4. NAT Traversal Scenarios

This section of the document includes detailed NAT traversal scenarios for both SIP signaling and the associated media.

4.1 Basic NAT SIP Signaling Traversal

The following sub-sections concentrate on SIP signaling traversal of NAT. The scenarios include traversal for both reliable and un-reliable transport protocols.

[Editors Note: The scenarios are still in early construction and a couple have been included as a hint of direction - All comments welcome for next release]

4.1.1 Registration (Registrar/Proxy Co-Located

The set of scenarios in this section document basic signaling traversal of a SIP REGISTER method through a NAT.

4.1.1.1 UDP

```
           Client              NAT               Proxy
             |                  |                  |
             |(1) REGISTER      |                  |
             |----------------->|                  |
             |                  |(1) REGISTER      |
             |                  |----------------->|
             |                  |(2) 401 Unauth    |
             |                  |<-----------------|
             |(2) 401 Unauth    |                  |
             |<-----------------|                  |
             |(3) REGISTER      |                  |
             |----------------->|                  |
             |                  |(3) REGISTER      |
             |                  |----------------->|
             |*************************************|
             |    Create Connection Re-use Tuple   |
             |*************************************|
             |                  |(4) 200 OK        |
             |                  |<-----------------|
             |(4) 200 OK        |                  |
             |<-----------------|                  |
             |                  |                  |
```

Figure 5.

Figure 5

In this example the client sends a SIP REGISTER request through a NAT which is challenged using the Digest authentication scheme. The client will include an 'rport' parameter as described in section 3.1.1 of this document for allowing traversal of UDP responses. The original request as illustrated in (1) in Figure 5 is a standard REGISTER message:

```
REGISTER sip:proxy.example.com SIP/2.0
Via: SIP/2.0/UDP client.example.com:5060;rport;branch=z9hG4bK
Max-Forwards: 70
Supported: gruu
From: Client <sip:client@example.com>;tag=djks8732
To: Client <sip:client@example.com>
Call-ID: 763hdc73y7dkb37@example.com
CSeq: 1 REGISTER
Contact: <sip:client@client.example.com>; connectioId=1
    ;+sip.instance="<urn:uuid:00000000-0000-0000-0000-000A95A0E120>"
Content-Length: 0
```

This proxy now generates a SIP 401 response to challenge for authentication, as depicted in (2) from Figure 5.:

```
    SIP/2.0 401 Unauthorized
    Via: SIP/2.0/UDP client.example.com:
5060;rport=8050;branch=z9hG4bK;received=192.0.1.2
    From: Client <sip:client@example.com>;tag=djks8732
    To: Client <sip:client@example.com>;tag=876877
    Call-ID: 763hdc73y7dkb37@example.com
    CSeq: 1 REGISTER
    WWW-Authenticate: [not shown]
    Content-Length: 0
```

The response will be sent to the address appearing in the 'received' parameter of the SIP 'Via' header (address 192.0.1.2). The response will not be sent to the port deduced from the SIP 'Via' header, as per standard SIP operation but will be sent to the value that has been stamped in the 'rport' parameter of the SIP 'Via' header (port 8050). For the response to successfully traverse the NAT, all of the conventions defined in RFC 3581 [5] MUST be obeyed. Make note of the both the 'connectionID' and 'sip.instance' contact header parameters. They are used to establish a connection re-use tuple as defined in [9]. The connection tuple creation is clearly shown in Figure 5. This ensures that any inbound request that causes a registration lookup will result in the re-use of the connection path established by the registration. This exonerates the need to manipulate contact header URI's to represent a globally routable address as perceived on the public side of a NAT. The subsequent messages defined in (3) and (4) from Figure 5 use the same mechanics for NAT traversal.

[Editors note: Will provide more details on heartbeat mechanism in next revision]

[Editors note: Can complete full flows if required on heartbeat inclusion]

4.1.1.2 Reliable Transport

```
           Client              NAT             Registrar
             |                  |                  |
             |(1) REGISTER      |                  |
             |----------------->|                  |
             |                  |(1) REGISTER      |
             |                  |----------------->|
             |                  |(2) 401 Unauth    |
             |                  |<-----------------|
             |(2) 401 Unauth    |                  |
             |<-----------------|                  |
             |(3) REGISTER      |                  |
             |----------------->|                  |
             |                  |(3) REGISTER      |
             |                  |----------------->|
             |*************************************|
             |    Create Connection Re-use Tuple   |
             |*************************************|
             |                  |(4) 200 OK        |
             |                  |<-----------------|
             |(4) 200 OK        |                  |
             |<-----------------|                  |
             |                  |                  |
```

Figure 6.

Traversal of SIP REGISTER messages request/responses using a reliable, connection orientated protocol such as TCP does not require any additional core SIP signaling extensions. SIP responses will re-use the connection created for the initial REGISTER request, (1) from Figure 6:

```
REGISTER sip:proxy.example.com SIP/2.0
Via: SIP/2.0/TCP client.example.com:5060;branch=z9hG4bKyilassjdshfu
Max-Forwards: 70
Supported: gruu
From: Client <sip:client@example.com>;tag=djks809834
To: Client <sip:client@example.com>
Call-ID: 763hdc783hcnam73@example.com
CSeq: 1 REGISTER
Contact: <sip:client@client.example.com;transport=tcp>; connectioId=1
    ;+sip.instance="<urn:uuid:00000000-0000-0000-0000-000A95A0E121>"
Content-Length: 0
```

This example was included to show the inclusion of the of the connection re-use Contact header parameters as defined in the Connection Re-use draft [9]. This creates an association tuple as described in the previous example for future inbound requests

directed at the newly created registration binding with the only difference that the association is with a TCP connection, not a UDP pin hole binding.

[Editors note: Will provide more details on heartbeat mechanism in next revision]

[Editors note: Can complete full flows on inclusion of heartbeat mechanism]

4.1.2 Registration(Registrar/Proxy not Co-Located)

This section demonstrates traversal mechanisms when the Registrar component is not co-located with the edge proxy element. The procedures described in this section are identical, regardless of transport protocol and so only one example will be documented in the form of TCP.

```
     Client              NAT                 Proxy             Registrar
       |                  |                   |                   |
       |(1) REGISTER      |                   |                   |
       |----------------->|                   |                   |
       |                  |(1) REGISTER       |                   |
       |                  |------------------>|                   |
       |                  |                   |(2) REGISTER       |
       |                  |                   |------------------>|
       |                  |                   |(3) 401 Unauth     |
       |                  |                   |<------------------|
       |                  |(4) 401 Unauth     |                   |
       |                  |<------------------|                   |
       |(4)401 Unauth     |                   |                   |
       |<-----------------|                   |                   |
       |(5)REGISTER       |                   |                   |
       |----------------->|                   |                   |
       |                  |(5)REGISTER        |                   |
       |                  |------------------>|                   |
       |                  |                   |(6)REGISTER        |
       |                  |                   |------------------>|
       |                  |                   |(7)200 OK          |
       |                  |                   |<------------------|
       |*********************************************************|
       |             Create Connection Re-use Tuple               |
       |*********************************************************|
       |                  |(8)200 OK          |                   |
       |                  |<------------------|                   |
       |(8)200 OK         |                   |                   |
       |<-----------------|                   |                   |
```

```
     |                  |                  |                  |
```

Figure 7.

This scenario builds on that contained in section 4.1.1.2. This time the REGISTER request is routed onwards to a separated Registrar. The important message to note is (5) in Figure 7. At this point, the proxy server routes the SIP REGISTER message to the Registrar. The proxy will create the connection re-use tuple at the same moment as the co-located example but for subsequent messages to arrive at the Proxy, the element needs to request to stay in the signaling path. REGISTER message (5) contains a SIP PATH extension header, as defined in RFC 3327 [6]. REGISTER message (5) would look as follows:

```
REGISTER sip:registrar.example.com SIP/2.0
Via: SIP/2.0/TCP proxy.example.com:5060;branch=z9hG4njkca8398hadjaa
Via: SIP/2.0/TCP client.example.com:5060;branch=z9hG4bKyilassjdshfu
Max-Forwards: 70
Supported: gruu
From: Client <sip:client@example.com>;tag=djks809834
To: Client <sip:client@example.com>
Call-ID: 763hdc783hcnam73@example.com
CSeq: 1 REGISTER
Path: <sip:sip%3Aclient%40example.com@proxy.example.com;lr>
Contact: <sip:client@client.example.com;transport=tcp>; connectioId=1
    ;+sip.instance="<urn:uuid:00000000-0000-0000-0000-000A95A0E121>"
Content-Length: 0
```

This results in the path header being stored along with the AOR and it's associated binding at the Registrar. The URI contained in the PATH will be inserted as a pre-loaded SIP 'Route' header into any request that arrives at the Registrar and is directed towards the associated binding. This guarantees that all requests for the new Registration will be forwarded to the edge proxy. The user part of the SIP 'Path' header URI that was inserted by the edge proxy contains an escaped form of the original AOR that was contained in the REGISTER request. On receiving subsequent requests, the edge proxy will examine the user part of the pre-loaded SIP 'route' header and extract the original AOR for use in it's connection tuple comparison, as defined in the connection re-use draft [9]. An example which will build on this scenario (showing an inbound request to the AOR) is detailed in section 4.1.4.2 of this document.

4.1.3 Initiating a Session

This section covers basic SIP signaling when initiating a call from

behind a NAT.

4.1.3.1 UDP

Initiating a call using UDP.

```
      Client              NAT               Proxy              [..]
        |                  |                  |
        |(1) INVITE        |                  |                 |
        |----------------->|                  |                 |
        |                  |(1) INVITE        |                 |
        |                  |----------------->|                 |
        |                  |(2) 407 Unauth    |                 |
        |                  |<-----------------|                 |
        |(2) 407 Unauth    |                  |                 |
        |<-----------------|                  |                 |
        |(3) INVITE        |                  |                 |
        |                  |(3) INVITE        |                 |
        |                  |----------------->|                 |
        |                  |                  |(4) INVITE       |
        |                  |                  |---------------->|
        |                  |                  |(5)180 RINGING   |
        |                  |                  |<----------------|
        |                  |(6)180 RINGING    |                 |
        |                  |<-----------------|                 |
        |(6)180 RINGING    |                  |                 |
        |<-----------------|                  |                 |
        |                  |                  |(7)200 OK        |
        |                  |                  |<----------------|
        |                  |(8)200 OK         |                 |
        |                  |<-----------------|                 |
        |(8)200 OK         |                  |                 |
        |<-----------------|                  |                 |
        |(9)ACK            |                  |                 |
        |----------------->|                  |                 |
        |                  |(9)ACK            |                 |
        |                  |----------------->|                 |
        |                  |                  |(10) ACK         |
        |                  |                  |---------------->|
        |                  |                  |(11)             |
```

Figure 8.

The initiating client generates an INVITE request that is to be sent through the NAT to a Proxy server. The INVITE message is represented in Figure 8 by (1) and is as follows:

```
INVITE sip:clientB@example.com SIP/2.0
Via: SIP/2.0/UDP client.example.com:5060;rport;branch=z9hG4bK74husdHG
Max-Forwards: 70
Route: <sip:proxy.example.com;lr>
From: clientA <sip:clientA@example.com>;tag=7skjdf38l
To: clientB <sip:clientB@example.com>
Call-ID: 8327468763423@example.com
CSeq: 1 INVITE
Contact: <sip:im_a_gruu@proxy.example.com>
Content-Type: application/sdp
Content-Length: ..

[SDP not shown]
```

There are a number of points to note with this message:

1. Firstly, as with the registration example in section 4.1.1.1, reponses to this request will not automatically pass back through a NAT and so the SIP 'Via' header 'rport' is included as described in the 'Symmetric response' section(3.1.1) and defined in RFC 3581 [5].
2. Secondly, the contact inserted contains the GRUU previously obtained from the registration.
3. [Editors Note: TODO - Expand description of GRUU and connection re-use]

4.1.3.2 Reliable Transport

[Editors note: TODO]

4.1.4 Receiving an Invitation to a Session

This section details scenarios where a client behind a NAT receives an inbound request through the NAT. These scenarios build on the previous registration scenario from sections 4.1.1 and 4.1.2 in this document.

4.1.4.1 Registrar/Proxy Co-located

The core SIP signaling associated with this call flow is not impacted directly by the transport protocol and so only one example scenario is necessary. The example uses UDP and follows on from the registration installed in the example from section 4.1.1.1.

```
       Client              NAT        Registrar/Proxy       SIP Entity
         |                  |                  |                  |
         |*******************************************************|
         |           Registration Binding Installed in           |
         |                    section 4.1.1.1                    |
         |*******************************************************|
         |                  |                  |                  |
         |                  |                  |(1)INVITE        |
         |                  |                  |<----------------|
         |                  |(2)INVITE         |                  |
         |                  |<-----------------|                  |
         |(2)INVITE         |                  |                  |
         |<-----------------|                  |                  |
         |                  |                  |                  |
         |                  |                  |                  |
```

Figure 9.

The core SIP signaling associated with this call flow is not impacted directly by the transport protocol and so only one example scenario is necessary. The example uses UDP and follows on from the registration installed in section 4.1.1.1. An INVITE request arrives at the Registrar with a destination pointing to the AOR of that inserted in section 4.1.1.1. The message is illustrated by (1) in Figure 9 and looks as follows:

```
INVITE sip:client@example.com SIP/2.0
Via: SIP/2.0/UDP external.example.com;branch=z9hG4bK74huHJ37d
Max-Forwards: 70
From: External <sip:External@external.example.com>;tag=7893hd
To: client <sip:client@example.com>
Call-ID: 8793478934897@external.example.com
CSeq: 1 INVITE
Contact: <sip:external@192.0.1.4>
Content-Type: application/sdp
Content-Length: ..

[SDP not shown]
```

The INVITE matches the registration binding at the Registrar and the INVITE request-URI is re-written to the selected onward address. The proxy then examines the request URI of the INVITE and compares with it's list of current open connections/mappings. It uses the incoming AOR to commence the check for associated open connections/mappings. Once matched, the proxy checks to see if the unique instance identifier (+sip.instance)associated with the binding equals the same

instance identifier associated with the binding. If more than one results are matched, the lowest 'connectionID' Contact parameter will be used. This is message (2) from Figure 9 and is as follows:

```
INVITE sip:sip:client@client.example.com SIP/2.0
Via: SIP/2.0/UDP proxy.example.com;branch=z9hG4kmlds893jhsd
Via: SIP/2.0/UDP external.example.com;branch=z9hG4bK74huHJ37d
Max-Forwards: 70
From: External <sip:External@external.example.com>;tag=7893hd
To: client <sip:client@example.com>
Call-ID: 8793478934897@external.example.com
CSeq: 1 INVITE
Contact: <sip:external@192.0.1.4>
Content-Type: application/sdp
Content-Length: ..

[SDP not shown]
```

It is a standard SIP INVITE request with no additional functionality. The major difference being that this request will not follow the address specified in the Request-URI, as standard SIP rules would enforce but will be sent on the connection/mapping associated with the registration binding. This then allows the original connection/mapping from the initial registration process to be re-used.

4.1.4.2 Registrar/Proxy Not Co-located

```
Client          NAT     Proxy      Registrar       SIP Entity
  |              |              |              |              |
  |***********************************************************|
  |            Registration Binding Installed in              |
  |                      section 4.1.2                        |
  |***********************************************************|
  |              |              |              |(1)INVITE     |
  |              |              |              |<-------------|
  |              |              |(2)INVITE     |              |
  |              |              |<-------------|              |
  |              |(3)INVITE     |              |              |
  |              |<-------------|              |              |
  |(3)INVITE     |              |              |              |
  |<-------------|              |              |              |
  |              |              |              |              |
  |              |              |              |              |
```

Figure 9.

4.2 Basic NAT Media Traversal

This section provides example scenarios to demonstrate basic media traversal using the techniques outlined earlier in this document.

4.2.1 Port Restricted Cone NAT

This section demonstrates an example of a client both initiating and receiving calls behind a 'Restricted Cone' NAT. The examples have been included to represent both 'Restricted' and 'Port Restricted' NAT media traversal. An example is included for both STUN and ICE with ICE being the RECOMENDED method.

4.2.1.1 STUN Solution

It is possible to traverse media through a 'Restricted Cone NAT' using STUN.

4.2.1.1.1 Initiating Session

The following example demonstrates media traversal through a 'Restricted Cone' NAT using STUN. It is assumed in this example that the STUN client and SIP Client are co-located on the same machine. Note that some SIP signalling messages have been left out for simplicity.

```
     Client               NAT                STUN               [..]
                                            Server
       |                  |                  |                  |
       |(1) STUN Req      |                  |                  |
       |src=10.0.1.1:5301 |                  |                  |
       |----------------->|                  |                  |
       |                  |(2) STUN Req      |                  |
       |                  |src=1.2.3.4:5601  |                  |
       |                  |----------------->|                  |
       |                  |(3) STUN Resp     |                  |
       |                  |<-----------------|                  |
       |                  |map=1.2.3.4:5601  |                  |
       |                  |dest=1.2.3.4:5601 |                  |
       |(4) STUN Resp     |                  |                  |
       |<-----------------|                  |                  |
       |map=1.2.3.4:5601  |                  |                  |
       |dest=10.0.1.1:5301|                  |                  |
```

```
       |(5) STUN Req      |                  |                  |
       |src=10.0.1.1:5302 |                  |                  |
       |----------------->|                  |                  |
       |                  |(6) STUN Req      |                  |
       |                  |src=1.2.3.4:5608  |                  |
       |                  |----------------->|                  |
       |                  |(7) STUN Resp     |                  |
       |                  |<-----------------|                  |
       |                  |map=1.2.3.4:5608  |                  |
       |                  |dest=1.2.3.4:5608 |                  |
       |(8) STUN Resp     |                  |                  |
       |<-----------------|                  |                  |
       |map=1.2.3.4:5608  |                  |                  |
       |dest=10.0.1.1:5302|                  |                  |
       |(9)SIP INVITE     |                  |                  |
       |----------------->|                  |                  |
       |                  |(10)SIP INVITE    |                  |
       |                  |------------------------------------>|
       |                  |                  |(11)SIP 200 OK    |
       |                  |<------------------------------------|
       |(12)SIP 200 OK    |                  |                  |
       |<-----------------|                  |                  |
       |========================================================|
       |>>>>>>>>>>>Outgoing Media sent to 1.2.3.4:5601>>>>>>>>>>|
       |========================================================|
       |========================================================|
       |<<<<<<<<<<<Incoming Media sent to 1.2.3.4:5601<<<<<<<<<<|
       |========================================================|
       |(13)SIP ACK       |                  |                  |
       |----------------->|                  |                  |
       |                  |(14) SIP ACK      |                  |
       |                  |------------------------------------>|
       |                  |                  |                  |
```

Figure 18: Restricted NAT with STUN - Initiating

- o On deciding to initiate a SIP voice session the VOIP client starts a local STUN client. The STUN client generates a standard STUN request as indicated in (1) from Figure 18 which also highlights the source address and port for which the client device wishes to obtain a mapping. The STUN request is sent through the NAT towards the public internet.
- o STUN message (2) traverses the NAT and breaks out onto the public internet towards the public STUN server. Note that the source address of the STUN requests now represents the public address and port from the public side of the NAT.

o
o The STUN server receives the request and processes it appropriately. This results in a successful STUN response being generated and returned (3). The message contains details of the mapped public address (contained in the STUN MAPPED-ADDRESS attribute) which is to be used by the originating client to receive media (see 'map=' from (3)).
o The STUN response traverses back through the NAT using the binding created by the STUN request and presents the new mapped address to the client (4). At this point the process is repeated to obtain a second mapped address (as shown in (5)-(8)) for an alternative local address (local port has now changed from 5301 to 5302 in (5)).
o The client now constructs a SIP INVITE message(9). Note that traversal of SIP is not covered in this example and is discussed in earlier sections of the document. The INVITE request will use the addresses it has obtained in the previous STUN transactions to populate the SDP of the SIP INVITE as shown below:

```
v=0
o=test 2890844526 2890842807 IN IP4 10.0.1.1
c=IN IP4 1.2.3.4
t=0 0
m=audio 5601 RTP/AVP 0
a=rtcp:5608
```

o Note that the mapped address obtained from the STUN transactions are inserted as the connection address for the SDP (c=1.2.3.4). The Primary port for RTP is also inserted in the SDP (m=audio 5601 RTP/AVP 0). Finally, the port gained from the additional STUN binding is placed in the RTCP attribute (as discussed in Section 3.2.5)for traversal of RTCP (a=rtcp:5608).
o The SIP signalling then traverses the NAT and sets up the SIP session (10-12). Note that the client transmits media as soon as the 200 OK to the INVITE arrives at the client(12). Up until this point the incoming media will not pass through the NAT as no outbound association has been created with the far end client. Two way media communication has now been established.

4.2.1.1.2 Receiving Session Invitation

Receiving a session for a 'Restricted Cone' NAT using STUN is very similar to the example outlined in Section 4.2.1.1.1. Figure 20 illustrates the associated flow of messages.

```
     Client              NAT                STUN               [..]
                                           Server
       |                  |                  | (1)SIP INVITE    |
       |                  |<-----------------|------------------|
       |(2) SIP INVITE    |                  |                  |
       |<-----------------|                  |                  |
       |                  |                  |                  |
       |(3) STUN Req      |                  |                  |
       |src=10.0.1.1:5301 |                  |                  |
       |----------------->|                  |                  |
       |                  |(4) STUN Req      |                  |
       |                  |src=1.2.3.4:5601  |                  |
       |                  |----------------->|                  |
       |                  |(5) STUN Resp     |                  |
       |                  |<-----------------|                  |
       |                  |map=1.2.3.4:5601  |                  |
       |                  |dest=1.2.3.4:5601 |                  |
       |(6) STUN Resp     |                  |                  |
       |<-----------------|                  |                  |
       |map=1.2.3.4:5601  |                  |                  |
       |dest=10.0.1.1:5301|                  |                  |
       |(7) STUN Req      |                  |                  |
       |src=10.0.1.1:5302 |                  |                  |
       |----------------->|                  |                  |
       |                  |(8) STUN Req      |                  |
       |                  |src=1.2.3.4:5608  |                  |
       |                  |----------------->|                  |
       |                  |(9) STUN Resp     |                  |
       |                  |<-----------------|                  |
       |                  |map=1.2.3.4:5608  |                  |
       |                  |dest=1.2.3.4:5608 |                  |
       |(10) STUN Resp    |                  |                  |
       |<-----------------|                  |                  |
       |map=1.2.3.4:5608  |                  |                  |
       |dest=10.0.1.1:5302|                  |                  |
       |(11)SIP 200 OK    |                  |                  |
       |----------------->|                  |                  |
       |                  |(12)SIP 200 OK    |                  |
       |                  |------------------------------------>|
       |========================================================|
       |>>>>>>>>>>>Outgoing Media sent to 1.2.3.4:5601>>>>>>>>>>|
       |========================================================|
       |========================================================|
       |<<<<<<<<<<<Incoming Media sent to 1.2.3.4:5601<<<<<<<<<<|
       |========================================================|
       |                  |                  |(13)SIP ACK       |
       |                  |<------------------------------------|
       |(14)SIP ACK       |                  |                  |
```

```
       |<-----------------|                  |                  |
       |                  |                  |                  |
```

Figure 20: Restricted NAT with STUN - Receiving

- o On receiving an invitation to a SIP voice session the VOIP client starts a local STUN client. The STUN client generates a standard STUN request as indicated in (3) from Figure 20 which also highlights the source address and port for which the client device wishes to obtain a mapping. The STUN request is sent through the NAT towards the public internet.
- o STUN message (4) traverses the NAT and breaks out onto the public internet towards the public STUN server. Note that the source address of the STUN requests now represents the public address and port from the public side of the NAT.
- o
- o The STUN server receives the request and processes appropriately. This results in a successful STUN response being generated and returned (5). The message contains details of the mapped public address (contained in the STUN MAPPED-ADDRESS attribute) which is to be used by the originating client to receive media (see 'map=' from (5)).
- o The STUN response traverses back through the NAT using the binding created by the STUN request and presents the new mapped address to the client (6). At this point the process is repeated to obtain a second mapped address (as shown in (7)-(10)) for an alternative local address (local port has now changed from 5301 to 5302 in (7)).
- o The client now constructs a SIP 200 OK message(11). Note that traversal of SIP is not covered in this example and is discussed in earlier sections of the document. The 200 OK response will use the addresses it has obtained in the previous STUN transactions to populate the SDP of the SIP INVITE as shown below:

```
v=0
o=test 2890844526 2890842807 IN IP4 10.0.1.1
c=IN IP4 1.2.3.4
t=0 0
m=audio 5601 RTP/AVP 0
a=rtcp:5608
```

- o Note that the mapped address obtained from the initial STUN transaction is inserted as the connection address for the SDP (c=1.2.3.4). The Primary port for RTP is also inserted in the SDP (m=audio 5601 RTP/AVP 0). Finally, the port gained from the additional binding is placed in the RTCP attribute (as discussed

in Section 3.2.5)for traversal of RTCP (a=rtcp:5608).

- o The SIP signalling then traverses the NAT and sets up the SIP session (11-14). Note that the client transmits media as soon as the 200 OK to the INVITE is sent to the UAC(11). Up until this point the incoming media will not pass through the NAT as no outbound association has been created with the far end client. Two way media communication has now been established.

4.2.1.2 ICE Solution

The preferred solution for media traversal of NAT is using ICE, as described in Section 3.2.4. The following examples illustrate the traversal of a 'Port Restricted Cone' NAT for both an initiating and receiving client. The example only covers ICE in association with STUN and TURN.

4.2.1.2.1 Initiating Session

The following example demonstrates an initiating traversal through a 'Restricted Cone' NAT using ICE.

```
 Client              NAT                STUN              TURN     [..]
                                       Server            Server
   |                  |                  |                  |        |
   |(1) STUN Req      |                  |                  |        |
   |src=10.0.1.1:5301 |                  |                  |        |
   |----------------->|                  |                  |        |
   |                  |(2) STUN Req      |                  |        |
   |                  |src=1.2.3.4:5601  |                  |        |
   |                  |----------------->|                  |        |
   |                  |(3) STUN Resp     |                  |        |
   |                  |<-----------------|                  |        |
   |                  |map=1.2.3.4:5601  |                  |        |
   |                  |dest=1.2.3.4:5601 |                  |        |
   |(4) STUN Resp     |                  |                  |        |
   |<-----------------|                  |                  |        |
   |map=1.2.3.4:5601  |                  |                  |        |
   |dest=10.0.1.1:5301|                  |                  |        |
   |(5) STUN Req      |                  |                  |        |
   |src=10.0.1.1:5311 |                  |                  |        |
   |----------------->|                  |                  |        |
   |                  |(6) STUN Req      |                  |        |
   |                  |src=1.2.3.4:5611  |                  |        |
   |                  |----------------->|                  |        |
   |                  |(7) STUN Resp     |                  |        |
   |                  |<-----------------|                  |        |
```

```
       |                  |map=1.2.3.4:5611  |                  |       |
       |                  |dest=1.2.3.4:5611 |                  |       |
       |(8) STUN Resp     |                  |                  |       |
       |<-----------------|                  |                  |       |
       |map=1.2.3.4:5611  |                  |                  |       |
       |dest=10.0.1.1:5311|                  |                  |       |
       |(9) TURN Allocate |                  |                  |       |
       |src=10.0.1.1:5302 |                  |                  |       |
       |----------------->|                  |                  |       |
       |                  |(10) TURN Allocate|                  |       |
       |                  |src=1.2.3.4:5608  |                  |       |
       |                  |------------------------------------>|       |
       |                  |(11) TURN Resp    |                  |       |
       |                  |<------------------------------------|       |
       |                  |map=1.2.3.4:5608  |                  |       |
       |                  |dest=1.2.3.4:5608 |                  |       |
       |(12) TURN Resp    |                  |                  |       |
       |<-----------------|                  |                  |       |
       |map=1.2.3.4:5608  |                  |                  |       |
       |dest=10.0.1.1:5302|                  |                  |       |
       |(13) TURN Allocate|                  |                  |       |
       |src=10.0.1.1:5312 |                  |                  |       |
       |----------------->|                  |                  |       |
       |                  |(14) TURN Allocate|                  |       |
       |                  |src=1.2.3.4:5618  |                  |       |
       |                  |------------------------------------>|       |
       |                  |(15) TURN Resp    |                  |       |
       |                  |<------------------------------------|       |
       |                  |map=1.2.3.4:5618  |                  |       |
       |                  |dest=1.2.3.4:5618 |                  |       |
       |(16) TURN Resp    |                  |                  |       |
       |<-----------------|                  |                  |       |
       |map=1.2.3.4:5618  |                  |                  |       |
       |dest=10.0.1.1:5312|                  |                  |       |
       |(17)SIP INVITE    |                  |                  |       |
       |----------------->|                  |                  |       |
       |                  |(18)SIP INVITE    |                  |       |
       |                  |-------------------------------------------->|
       |                  |                  |(19)SIP 200 OK    |       |
       |                  |<--------------------------------------------|
       |(20)SIP 200 OK    |                  |                  |       |
       |<-----------------|                  |                  |       |
       |(21)STUN Req      |                  |                  |       |
       |----------------->|                  |                  |       |
       |                  |(22) STUN Req     |                  |       |
       |                  |-------------------------------------------->|
       |                  |                  |(23)STUN Resp     |       |
       |                  |<--------------------------------------------|
```

```
       |(24)STUN Resp     |                  |                  |       |
       |<-----------------|                  |                  |       |
       |================================================================|
       |>>>>>>>>>>>Outgoing Media sent from 10.1.1.1:5301>>>>>>>>>>>>>>>|
       |================================================================|
       |                  |                  |(25) STUN Req     |       |
       |                  |<--------------------------------------------|
       |(26)STUN Req      |                  |                  |       |
       |<-----------------|                  |                  |       |
       |(27)STUN Resp     |                  |                  |       |
       |----------------->|                  |                  |       |
       |                  |                  |(28)STUN Resp     |       |
       |                  |-------------------------------------------->|
       |================================================================|
       |<<<<<<<<<<<Incoming Media sent to 1.2.3.4:5601<<<<<<<<<<<<<<<<<<|
       |================================================================|
       |(29)SIP ACK       |                  |                  |       |
       |----------------->|                  |                  |       |
       |                  |(30) SIP ACK      |                  |       |
       |                  |-------------------------------------------->|
       |                  |                  |                  |       |
```

Figure 22: Restricted NAT with ICE - Initiating

o On deciding to initiate a SIP voice session the VOIP client starts a local STUN and TURN client. The STUN client generates a standard STUN request as indicated in (1) from Figure 22 which also highlights the source address and port for which the client device wishes to obtain a mapping. The STUN request is sent through the NAT towards the public internet.

o STUN message (2) traverses the NAT and breaks out onto the public internet towards the public STUN server. Note that the source address of the STUN requests now represents the public address and port from the public side of the NAT.

o

o The STUN server receives the request and processes appropriately. This results in a successful STUN response being generated and returned (3). The message contains details of the mapped public address (contained in the STUN MAPPED-ADDRESS attribute) which is to be used by the originating client to receive media (see 'map=') from (3)).

o The STUN response traverses back through the NAT using the binding created by the STUN request and presents the new mapped address to the client (4). The process is repeated and a second STUN derived address is obtained, as illustrated in (5)-(8) in Figure 22. While the STUN client is obtaining addresses', the TURN client will also be attempting to obtain external representations. The

TURN Allocate message is constructed in association with the local IP address and port combination(9). The TURN Allocate message is then sent from the client to the external TURN server via the NAT(10). The TURN server processes the Allocate request and returns an appropriate response(11). The response contains the 'Mapped-Address'(defined in STUN specification) attribute which contains the external representation that the TURN server will provide for the internal mapping. The TURN response then traverses back through NAT and returns the newly allocated external representation to the originating client(12).The process is repeated and a second TURN derived address is obtained, as illustrated in (13)-(16) in Figure 22. At this point the client behind the NAT has a pair of STUN external representations and TURN equivalents. The client would be free to gather any number of external representations using any UNSAF[7] compliant protocol.

- The client now constructs a SIP INVITE message(17). The INVITE request will use the addresses it has obtained in the previous STUN/TURN interactions to populate the SDP of the SIP INVITE. This should be carried out in accordance with the semantics defined in the ICE specification[13], as shown below (*note - /* signifies line continuation):

```
v=0
o=test 2890844526 2890842807 IN IP4 10.0.1.1
c=IN IP4 1.2.3.4
t=0 0
m=audio 5601 RTP/AVP 0
a=candidate:H83jksd 1.0 rtp_uname_frag_1 rtp_pass_1 1.2.3.4 5601
  /* rtcp_uname_frag_1 rtcp_pass_1 1.2.3.4 5611
a=candidate:Hye73hd 0.8 rtp_uname_frag_2 rtp_pass_2 1.2.3.4 5608
  /* rtcp_uname_frag_2 rtcp_pass_2 1.2.3.4 5618
a=candidate:H82hjjh 0.5 rtp_uname_frag_3 rtp_pass_3 1.2.3.4 5600
```

- The SDP has been constructed to include all the available addresses that have been assembled. The first 'candidate' address contains the two STUN derived addresses for both RTP and RTCP traffic. This entry has been given the highest priority(1.0) by the client and also inserted as the default address.
- The second 'candidate' address contains the two TURN derived addresses for both RTP and RTCP traffic. This entry has been given the second highest priority(0.8).
- The third and final 'candidate' address contains a local interface address that has not been derived externally. This entry has been given the lowest priority(0.5).
- The SIP signalling then traverses the NAT and sets up the SIP session (18)-(20). On advertising a candidate address, the client should have a local STUN server running on each advertised

candidate address. This is for the purpose of responding to incoming connectivity checks. In this example, after sending the INVITE and receiving a 200 OK, the client initiates an outgoing STUN connectivity check to the selected remote interfaces (21)-(24) (*Note - this process will be repeated for every advertised address which is not shown in the diagram for simplicity). On receiving a STUN response, the client is able to stream media to the remote destination(*Note - if further STUN connectivity responses are received after the client has started streaming media with a higher priority, it will be used instead). The remote destination will also carry out similar STUN connectivity checks (25)-(28) which then allows media to be streamed to the client behind the NAT using the advertised connections. Two way audio is now possible between the two clients.

4.2.1.2.2 Receiving Session Invitation

This example is similar to that described in Section 4.2.1.2.1. The client behind a NAT is receiving the incoming ICE Initiate in a SIP INVITE request.

```
     Client              NAT               STUN               TURN     [..]
                                          Server             Server
       |                  |                  |                  |       |
       |                  |                  |(1)SIP INVITE     |       |
       |                  |<--------------------------------------------|
       |(2)SIP INVITE     |                  |                  |       |
       |<-----------------|                  |                  |       |
       |                  |                  |                  |       |
       |(3) STUN Req      |                  |                  |       |
       |src=10.0.1.1:5301 |                  |                  |       |
       |----------------->|                  |                  |       |
       |                  |(4) STUN Req      |                  |       |
       |                  |src=1.2.3.4:5601  |                  |       |
       |                  |----------------->|                  |       |
       |                  |(5) STUN Resp     |                  |       |
       |                  |<-----------------|                  |       |
       |                  |map=1.2.3.4:5601  |                  |       |
       |                  |dest=1.2.3.4:5601 |                  |       |
       |(6) STUN Resp     |                  |                  |       |
       |<-----------------|                  |                  |       |
       |map=1.2.3.4:5601  |                  |                  |       |
       |dest=10.0.1.1:5301|                  |                  |       |
       |(7) STUN Req      |                  |                  |       |
       |src=10.0.1.1:5311 |                  |                  |       |
```

```
       |----------------->|                  |                  |       |
       |                  |(8) STUN Req      |                  |       |
       |                  |src=1.2.3.4:5611  |                  |       |
       |                  |----------------->|                  |       |
       |                  |(9) STUN Resp     |                  |       |
       |                  |<-----------------|                  |       |
       |                  |map=1.2.3.4:5611  |                  |       |
       |                  |dest=1.2.3.4:5611 |                  |       |
       |(10) STUN Resp    |                  |                  |       |
       |<-----------------|                  |                  |       |
       |map=1.2.3.4:5611  |                  |                  |       |
       |dest=10.0.1.1:5311|                  |                  |       |
       |(11) TURN Allocate|                  |                  |       |
       |src=10.0.1.1:5302 |                  |                  |       |
       |----------------->|                  |                  |       |
       |                  |(12) TURN Allocate|                  |       |
       |                  |src=1.2.3.4:5608  |                  |       |
       |                  |------------------------------------>|       |
       |                  |(13) TURN Resp    |                  |       |
       |                  |<------------------------------------|       |
       |                  |map=1.2.3.4:5608  |                  |       |
       |                  |dest=1.2.3.4:5608 |                  |       |
       |(14) TURN Resp    |                  |                  |       |
       |<-----------------|                  |                  |       |
       |map=1.2.3.4:5608  |                  |                  |       |
       |dest=10.0.1.1:5302|                  |                  |       |
       |(15) TURN Allocate|                  |                  |       |
       |src=10.0.1.1:5312 |                  |                  |       |
       |----------------->|                  |                  |       |
       |                  |(16) TURN Allocate|                  |       |
       |                  |src=1.2.3.4:5618  |                  |       |
       |                  |------------------------------------>|       |
       |                  |(17) TURN Resp    |                  |       |
       |                  |<------------------------------------|       |
       |                  |map=1.2.3.4:5618  |                  |       |
       |                  |dest=1.2.3.4:5618 |                  |       |
       |(18) TURN Resp    |                  |                  |       |
       |<-----------------|                  |                  |       |
       |map=1.2.3.4:5618  |                  |                  |       |
       |dest=10.0.1.1:5312|                  |                  |       |
       |(19)SIP 200 OK    |                  |                  |       |
       |----------------->|                  |                  |       |
       |                  |(20)SIP 200 OK    |                  |       |
       |                  |-------------------------------------------->|
       |(21)STUN Req      |                  |                  |       |
       |----------------->|                  |                  |       |
       |                  |(22) STUN Req     |                  |       |
       |                  |-------------------------------------------->|
```

```
       |                  |                  |(23)STUN Resp     |        |
       |                  |<-------------------------------------------|
       |(24)STUN Resp     |                  |                  |        |
       |<-----------------|                  |                  |        |
       |=================================================================|
       |>>>>>>>>>>>Outgoing Media sent from 10.1.1.1:5301>>>>>>>>>>>>>>>|
       |=================================================================|
       |                  |                  |(25) STUN Req     |        |
       |                  |<-------------------------------------------|
       |(26)STUN Req      |                  |                  |        |
       |<-----------------|                  |                  |        |
       |(27)STUN Resp     |                  |                  |        |
       |----------------->|                  |                  |        |
       |                  |                  |(28)STUN Resp     |        |
       |                  |------------------------------------------->|
       |=================================================================|
       |<<<<<<<<<<<Incoming Media sent to 1.2.3.4:5601<<<<<<<<<<<<<<<<<<|
       |=================================================================|
       |                  |                  |(29)SIP ACK       |        |
       |                  |<-------------------------------------------|
       |(30)SIP ACK       |                  |                  |        |
       |<-----------------|                  |                  |        |
       |                  |                  |                  |        |
```

Figure 24: Restricted NAT with ICE - Receiving

- o As mentioned previously, this example is similar that described in Section 4.2.1.2.1. For this reason, some of the description may reference the previous example. The scenario starts with the client behind the NAT receiving a SIP INVITE(1) request(ICE initiate message).
- o On receiving the SIP INVITE the client is able to collect all possible addresses available for media interaction (e.g. Local addresses, STUN derived, TURN derived). See detail from Section 4.2.1.2.1 for explanation on accumulating all possible media addresses (Steps (3)-(18) in Figure 24).
- o The client will perform connectivity checks on all addresses received in the SIP INVITE message(21)-(24). Note that steps (21)-(24) will be repeated for every address offered in the SIP INVITE request. This is not shown in the diagram for simplicity. On receiving a response to a STUN connectivity check, the client will start streaming media(*Note - if further STUN connectivity responses are received after the client has started streaming meda with a higher priority, it will be used instead).
- o The STUN connectivity checks will then occur in the opposite direction, as illustrated in Section 4.2.1.2.1. A STUN server running on each advertised address will respond to incoming STUN

connectivity requests(25)-(28).

o Bi-directional audio can now occur between the two clients.

4.2.2 Symmetric NAT

4.2.2.1 STUN Failure

This section highlights that while STUN is the preferred mechanism for traversal of NAT, it does not solve every cases. The use of STUN on its own will not guarantee traversal through every NAT type, hence the recommendation that ICE be the preffered option.

```
  Client             SYMMETRIC           STUN                [..]
                       NAT              Server
     |                  |                  |                  |
     |(1) STUN Req      |                  |                  |
     |src=10.0.1.1:5301 |                  |                  |
     |----------------->|                  |                  |
     |                  |(2) STUN Req      |                  |
     |                  |src=1.2.3.4:5601  |                  |
     |                  |----------------->|                  |
     |                  |(3) STUN Resp     |                  |
     |                  |<-----------------|                  |
     |                  |map=1.2.3.4:5601  |                  |
     |                  |dest=1.2.3.4:5601 |                  |
     |(4) STUN Resp     |                  |                  |
     |<-----------------|                  |                  |
     |map=1.2.3.4:5601  |                  |                  |
     |dest=10.0.1.1:5301|                  |                  |
     |(5)SIP INVITE     |                  |                  |
     |----------------->|                  |                  |
     |                  |(6)SIP INVITE     |                  |
     |                  |------------------------------------>|
     |                  |                   |(7)SIP 200 OK    |
     |                  |<------------------------------------|
     |(8)SIP 200 OK     |                  |                  |
     |<-----------------|                  |                  |
     |=======================================================|
     |>>>>>>>>>>>Outgoing Media sent from 10.0.1.1:5301>>>>>>>|
     |=======================================================|
     |                  x====================================|
     |                  xIncoming Media sent to 1.2.3.4:5601<<|
     |                  x====================================|
     |(9)SIP ACK        |                  |                  |
     |----------------->|                  |                  |
     |                  |(10) SIP ACK      |                  |
```

```
       |                  |----------------------------------->|
       |                  |                  |                 |
```

Figure 25: Symmetric NAT with STUN - Failure

The example in Figure 25 is conveyed in the context of the client behind the Symmetric NAT initiating a call. It should be noted that the same problem applies when a client receives a SIP invitation and is behind a Symmetric NAT.

- o In Figure 25 the client behind the NAT obtains an external representation using standard STUN mechanisms (1)-(4) that have been used in previous examples in this document (e.g Section 4.2.1.1.1).
- o The external mapped address obtained is also used in the outgoing SDP contained in the SIP INVITE request(5).
- o In this example the client is still able to send media to the external client. The problem occurs when the client outside the NAT tries to use the address supplied in the outgoing INVITE request to traverse media back through the Symmetric NAT.
- o A symmetric NAT has differing rules from the Cone variety of NAT. For any internal IP address and port mapping, data sent to different external addresses does not provide the same public mapping at the NAT. In Figure 25 the STUN query produced a valid external mapping. This mapping, however, can only be used in the context of the original STUN request that was sent to the STUN server. Any packets that attempt to use the mapped address, that does not come from the STUN server IP address and port, will be dropped at the NAT. Figure 25 shows the media being dropped at the NAT after (8).

4.2.2.2 TURN Solution

As identified in Section 4.2.2.1, STUN provides a useful tool for the traversal of the majority of NATs but fails with symmetric type NAT. This led to the development of the TURN solution[11] which introdcues a media relay in the path for NAT traversal (as described in Section 3.2.3). The following example explains how TURN solves the previous failure when using STUN to traverse a symmetric NAT.

```
     Client            SYMMETRIC            TURN                 [..]
                         NAT               Server
       |                  |                  |                  |
       |(1) TURN Allocate |                  |                  |
       |src=10.0.1.1:5301 |                  |                  |
       |----------------->|                  |                  |
```

```
       |                  |(2) TURN Allocate |                  |
       |                  |src=1.2.3.4:5601  |                  |
       |                  |----------------->|                  |
       |                  |(3) TURN Resp     |                  |
       |                  |<-----------------|                  |
       |                  |map=2.3.4.5:5601  |                  |
       |                  |dest=1.2.3.4:5601 |                  |
       |(4) TURN Resp     |                  |                  |
       |<-----------------|                  |                  |
       |map=2.3.4.5:5601  |                  |                  |
       |dest=10.0.1.1:5301|                  |                  |
       |(5)SIP INVITE     |                  |                  |
       |----------------->|                  |                  |
       |                  |(6)SIP INVITE     |                  |
       |                  |------------------------------------>|
       |                  |                   |(7)SIP 200 OK    |
       |                  |<------------------------------------|
       |(8)SIP 200 OK     |                  |                  |
       |<-----------------|                  |                  |
       |========================================================|
       |>>>>>>>>>>>Outgoing Media sent from 10.0.1.1:5301>>>>>>>|
       |========================================================|
       |                  |                  |==================|
       |                  |                  |<<<Media Sent to<<|
       |                  |                  |<<< 2.3.4.5:5601<<|
       |                  |                  |==================|
       |=====================================|                  |
       |<Incoming Media Sent to 1.2.3.4:5601<|                  |
       |=====================================|                  |
       |(9)SIP ACK        |                  |                  |
       |----------------->|                  |                  |
       |                  |(10) SIP ACK      |                  |
       |                  |------------------------------------>|
       |                  |                  |                  |
```

Figure 26: Symmetric NAT with STUN - Failure

o The client obtains a TURN derived address by issuing TURN allocate request(1). The request traverses through the symmetric NAT and reaches the TURN server (2). The Turn server generates a response that contains an external representation. The representation maps to an address mapping on the TURN server which is bound to the public pin hole in the NAT, opened by the TURN request. This results in any traffic being sent to the TURN server representation (2.3.4.5:5601) will be redirected to the external representation of the pin hole created by the TURN request(1.2.3.4:5601).

o The TURN derived address (2.3.4.5:5601) arrives back at the originating client(4). This address can then be used in the SDP for the outgoing SIP INVITE request as shown below (note that the RTCP attribute would have been obtained by another TURN derived address which is not shown in the call flow for simplicity):-

o

```
v=0
o=test 2890844342 2890842164 IN IP4 10.0.1.1
c=IN IP4 2.3.4.5
t=0 0
m=audio 5601 RTP/AVP 0
a=rtcp:5608
```

o On receiving the INVITE request, the UAS is able to stream media to the TURN derived address (2.3.4.5:5601). As shown in Figure 26, the media from the UAS is directed to the TURN derived address at the TURN server. The TURN server then redirects the traffic to the open pin hole in the symmetric NAT(1.2.3.4:5601). The media traffic is then able to traverse the symmetric NAT and arrives back at the client.

o The TURN solution on its own will work for Symmetric and other types of NAT mentioned in this specification but should only be used as a last resort. The relaying of media through an external entity is not an efficient mechanism for all NAT traversal.

4.2.2.3 ICE Solution

The previous two examples have highlighted the problem with using STUN for all forms of NAT traversal and a solution using TURN for the symmetric NAT case. As mentioned previously in this document, the RECOMMENDED mechanism for traversing all varieties of NAT is using ICE, as detailed in Section 3.2.4. Ice maked use of STUN, TURN and any other UNSAF [7] compliant protocol to provide a list of prioritised addresses that can be used for media traffic. Detailed examples of ICE can be found in Section 4.2.1.2.1 and in Section 4.2.1.2.2. These examples are associated with a 'Port Restricted' type NAT but can be applied to any NAT type variation, including 'Symmetric' type NAT. The procedures are the same and of the list of candidate addresses, a client will choose where to send media dependant on the results of the STUN connectivity checks on each candidate address and the associated priority (highest priority wins). For more information see the core ICE specificagtion[13]

4.3 Advanced NAT media Traversal Using ICE

4.3.1 Full Cone --> Full Cone traversal

4.3.1.1 Without NAT

4.3.1.1.1 Initiating Session

4.3.1.1.2 Receiving Session Invitation

4.3.1.2 With NAT

4.3.1.2.1 Initiating Session

4.3.1.2.2 Receiving Session Invitation

4.3.2 Port Restricted Cone --> Port Restricted Cone traversal

4.3.2.1 Without NAT

4.3.2.1.1 Initiating Session

4.3.2.1.2 Receiving Session Invitation

4.3.2.2 With NAT

4.3.2.2.1 Initiating Session

4.3.2.2.2 Receiving Session Invitation

4.3.3 Internal TURN Server (Enterprise Deployment)

4.3.3.1 Peer in same Enterprise

4.3.3.2 Peer in same Enterprise - Separated by NAT

4.3.3.3 Peer outside Enterprise

4.4 Intercepting Intermediary (B2BUA)

4.5 IPV4/IPV6

4.6 ICE with RTP/TCP

5. References

5.1 Normative References

[1] 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.

[2] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

[3] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

[4] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[5] Rosenberg, J. and H. Schulzrinne, "An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing", RFC 3581, August 2003.

[6] Willis, D. and B. Hoeneisen, "Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts", RFC 3327, December 2002.

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

[8] 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.

[9] Jennings, C. and A. Hawrylyshen, "SIP Conventions for Connection Usage", Internet-Draft draft-jennings-sipping-outbound-00, October 2004.

[10] Rosenberg, J., "Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)", Internet-Draft draft-ietf-sip-gruu-02, July 2004.

[11] Rosenberg, J., "Traversal Using Relay NAT (TURN)", Internet-Draft draft-rosenberg-midcom-turn-06, October 2004.

[12] Wing, D., "Symmetric RTP and RTCP Considered Helpful", Internet-Draft draft-wing-mmusic-symmetric-rtprtcp-01, October 2004.

[13] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Multimedia Session Establishment Protocols", Internet-Draft draft-ietf-mmusic-ice-03, October 2004.

[14] Rosenberg, J., "Simple Traversal of UDP Through Network Address Translators (NAT) (STUN)", Internet-Draft draft-ietf-behave-rfc3489bis-00, October 2004.

5.2 Informative References

Authors' Addresses

Chris Boulton
Ubiquity Software Corporation
Eastern Business Park
St Mellons
Cardiff, South Wales CF3 5EA

Email: cboulton@ubiquitysoftware.com

Jonathan Rosenberg
Cisco Systems
600 Lanidex Plaza
Parsippany, NJ 07054

Email: jdrosen@dynamicsoft.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.