SIPPING Working Group
Internet-Draft
Expires: December 28, 2006

C. Boulton, Ed.
Ubiquity Software Corporation
J. Rosenberg
Cisco Systems
G. Camarillo
Ericsson
June 26, 2006

Best Current Practices for NAT Traversal for SIP draft-ietf-sipping-nat-scenarios-05

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 10
3.2.4. ICE 10
3.2.5. 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 18
4.1.4. Receiving an Invitation to a Session 20
4.2. Basic NAT Media Traversal 23
4.2.1. Endpoint independent NAT 24
4.2.2. Port and Address Dependant NAT 40
4.3. Address independent Port Restricted NAT --> Address independent Port Restricted NAT traversal 46
4.4. Internal TURN Usage (Enterprise Deployment) 46
5. Intercepting Intermediary (B2BUA) 46
6. IPv4-IPv6 Transition 47
6.1. IPv4-IPv6 Transition for SIP Signalling 47
6.2. IPv4-IPv6 Transition for Media 47
7. ICE with RTP/TCP 50
8. Acknowledgments 50
9. References 50
9.1. Normative References 50
9.2. Informative References 52
Authors' Addresses 53
Intellectual Property and Copyright Statements 54

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 behaviors can be found in 'NAT Behavioral Requirements for Unicast UDP' [11].

The IETF has produced many specifications for the traversal of NAT, including STUN, ICE, rport, symmetric RTP, TURN, SIP Outbound, 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 its 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 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.

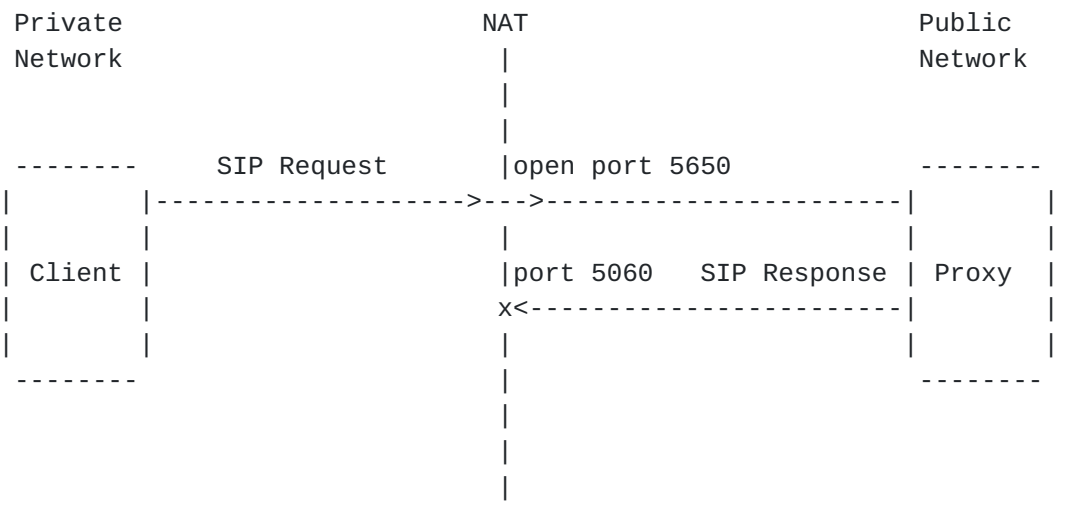


Figure 1

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 reverse direction of the originating client 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.

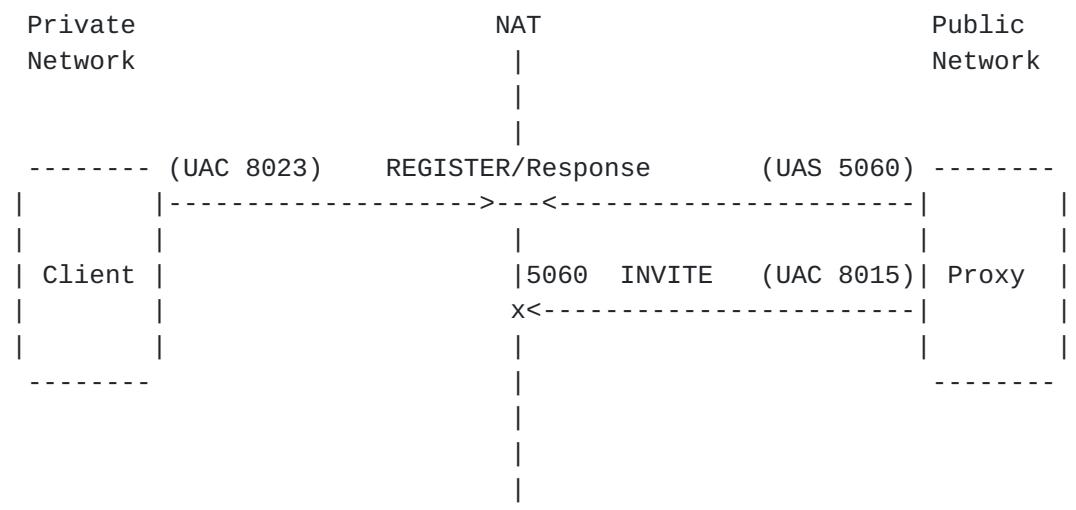


Figure 2

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 the inactivity timer, and 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 resources.

Media associated with SIP calls also has problems traversing NAT. RTP [2]] is one of the most common media transport types 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[5]. 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 its address to receive media and it exists in a private network that is not accessible from outside the NAT. Figure 3 illustrates this problem.

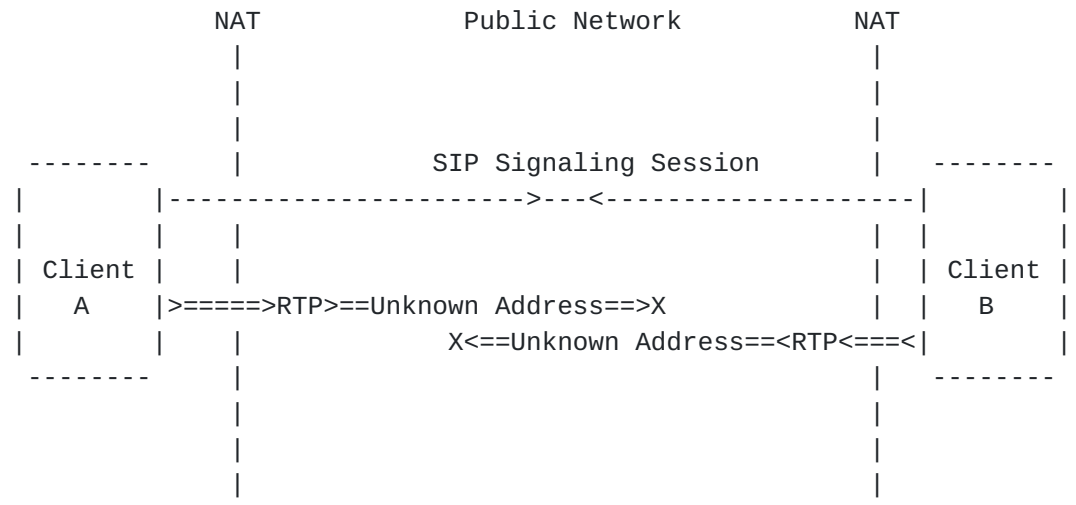


Figure 3

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 reaches the public internet and is not resolvable. The media traffic fails. 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[5]. 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.

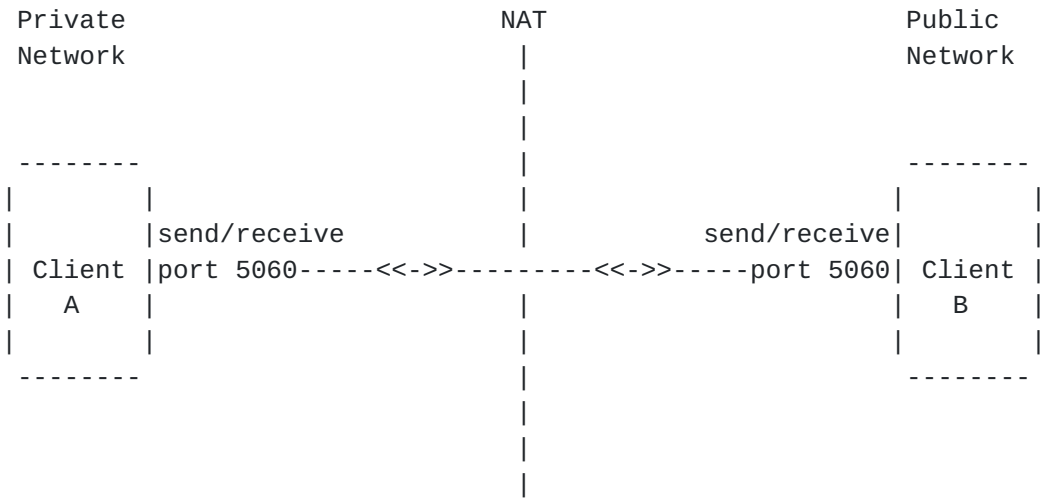


Figure 4

The exact functionality for this method of response traversal is

called 'Symmetric Response' and the details are documented in RFC 3581 [6]. 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 2 and illustrated in Figure 2, is to allow incoming requests to be properly routed.

Guidelines for devices such as User Agents that can only generate outbound connections through a NAT are documented in 'SIP Conventions for UAs with Outbound Only Connections'[13]. The document provides techniques that use a unique User Agent instance identifier (instance-id) in association with a flow identifier (Reg-id). The combination of the two identifiers provides a key to a particular connections (both UDP and TCP) that are stored in association with registration bindings. On receiving an incoming request to a SIP Address-Of-Record (AOR), a proxy routes to the associated flow created by the registration and thus a route through a NAT. It also provides a keepalive mechanism for clients to keep NAT bindings alive. This is achieved using peer-to-peer STUN multiplexed over the SIP signaling connection. Usage of this specification is RECOMMENDED. This mechanism is not transport specific and should be used for any transport protocol.

Even if the SIP Outbound draft is not used, clients generating SIP requests 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 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 [15]. '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.

3.2.1.1. 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 [6] 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).

3.2.2. STUN

Simple Traversal of User Datagram Protocol (UDP) through Network Address Translators(NAT) or STUN is defined in RFC 3489bis [10]. STUN is a lightweight tool kit and protocol that 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 it accordingly as the connection address in SDP to provide NAT traversal. Using terminology defined in the draft 'NAT Behavioral Requirements for Unicast UDP' [11], STUN does work with 'Endpoint Independent Mapping' but does not work with either 'Address Dependent Mapping' or 'Address Dependent and Port Mapping' type NATs. Using STUN with either of the previous two NAT mappings to probe for the external IP address/port representation 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.

As mentioned in Section 3.1.2, STUN is also used as a peer-to-peer

keep-alive mechanism.

3.2.3. TURN

As described in the previous section, the STUN protocol does not work for UDP traversal through certain identified NAT mappings. 'Obtaining Relay Addresses from Simple Traversal of UDP Through NAT (known as TURN)' is a usage of the STUN protocol for deriving (from a STUN server) an address that will be used to relay packet towards a client. TURN provides an external address (globally routable) at a STUN server that will act as a media relay which guarantees traffic will reach the associated internal address. The full details of the TURN specification are defined in [12]. 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 and can lead to unwanted latency.

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[9] 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 [16].

3.2.5. 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.

3.2.5.1. Primary Profile

A client falling into the 'Primary' profile supports ICE in conjunction with STUN, TURN and RFC 3605 [6] 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 varieties of NAT.

3.2.5.2. Consumer Profile

A client falling into the 'Consumer' profile supports STUN and RFC 3605 [6] 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.

3.2.5.3. 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.

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

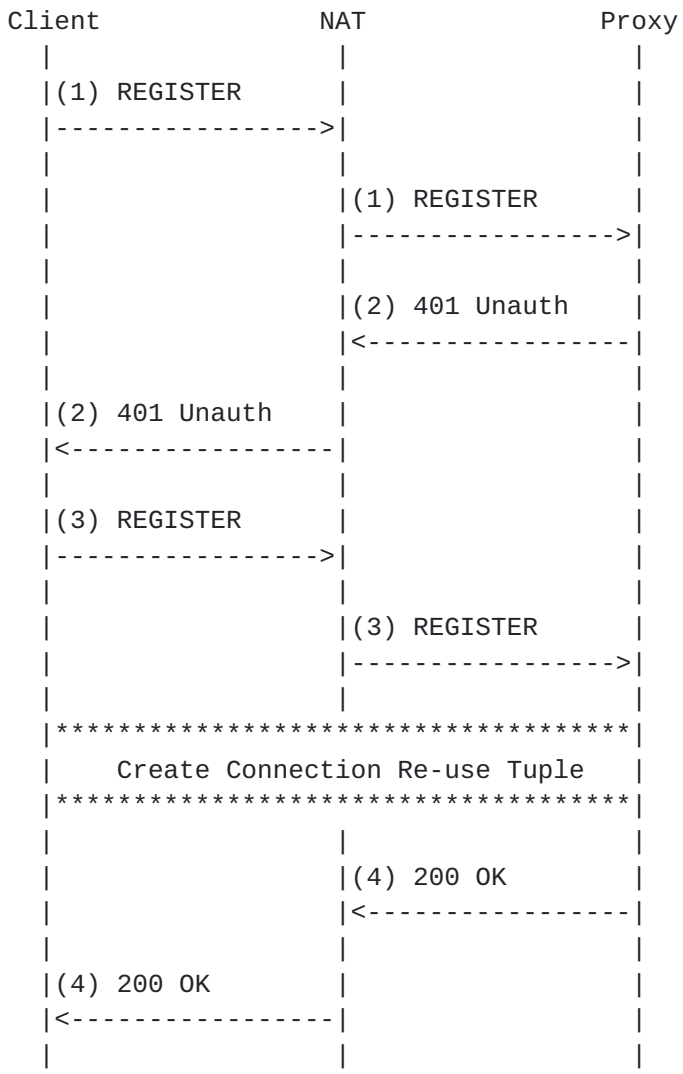


Figure 5

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: path,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>;reg-id=1
     ;+sip.instance="<urn:uuid:00000000-0000-0000-0000-00A95A0E120>"
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 [6] 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 [13]. 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

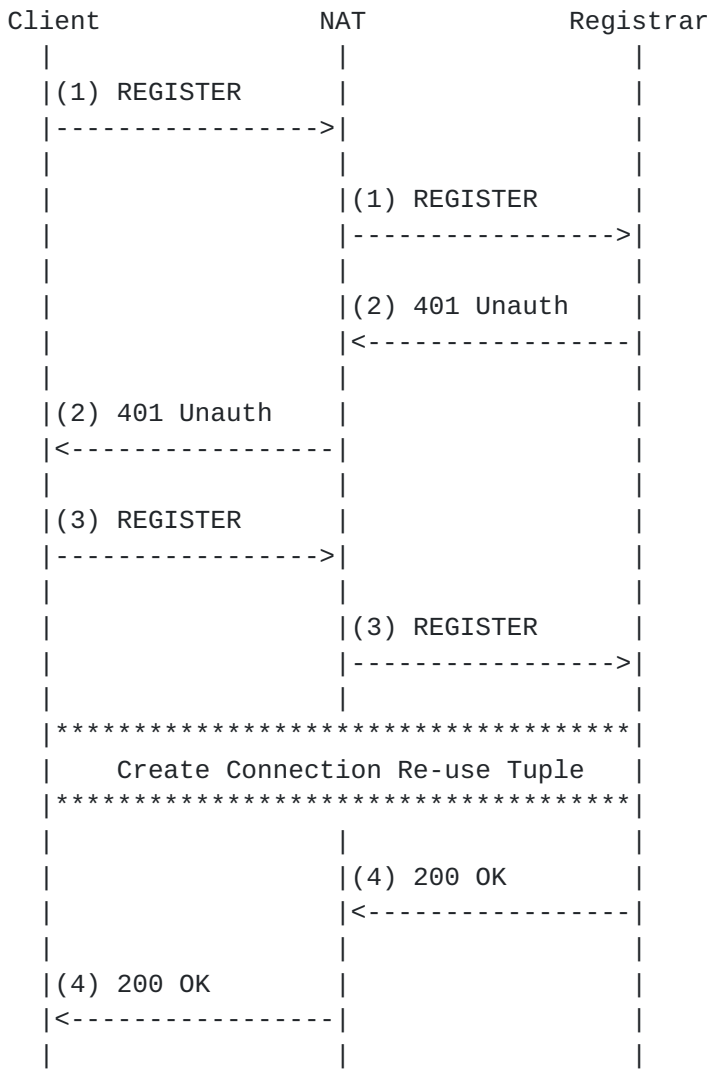


Figure 6.

Traversal of SIP REGISTER requests/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: path,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>;reg-id=1
     ;+sip.instance="<urn:uuid:00000000-0000-0000-0000-00A95A0E121>"
Content-Length: 0
```

This example was included to show the inclusion of the connection re-use Contact header parameters as defined in the Connection Re-use draft [13]. 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.

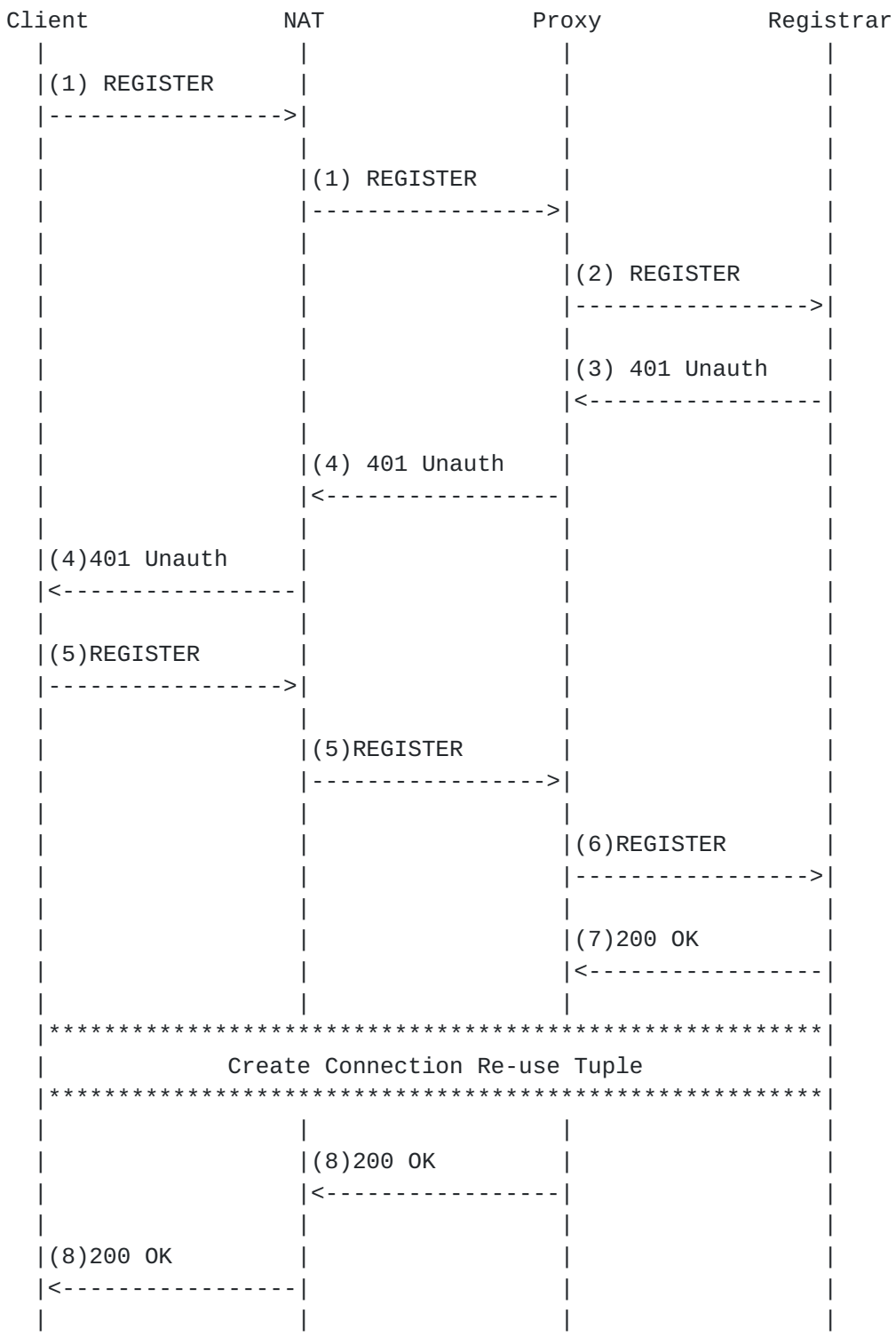


Figure 7.

This scenario builds on the previous example contained in Section 4.1.1.2. The primary difference being that the REGISTER request is routed onwards from a Proxy Server to a separated

Registrar. The important message to note is (6) in Figure 7. The Edge proxy, on receiving a REGISTER request that contains a 'sip.instance' media feature tag, forms a unique flow identifier token as discussed in [13] . At this point, the proxy server routes the SIP REGISTER message to the Registrar. The proxy will create the connection tuple as described in SIP Outbound at the same moment as the co-located example, but for subsequent messages to arrive at the Proxy, the element needs to request to remain in the signaling path. To achieve this the proxy inserts to REGISTER message (5) a SIP PATH extension header, as defined in RFC 3327 [7]. The previously created flow token is inserted in a position within the Path header where it can easily be retrieved at a later point when receiving messages to be routed to the registration binding. 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: path,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:3HS28o8HAKJSH&&U@proxy.example.com;lr>
 Contact: <sip:client@client.example.com;transport=tcp>;
      ;+sip.instance="<urn:uuid:00000000-0000-0000-0000-00A95A0E121>";reg-
id=1
 Content-Length: 0
```

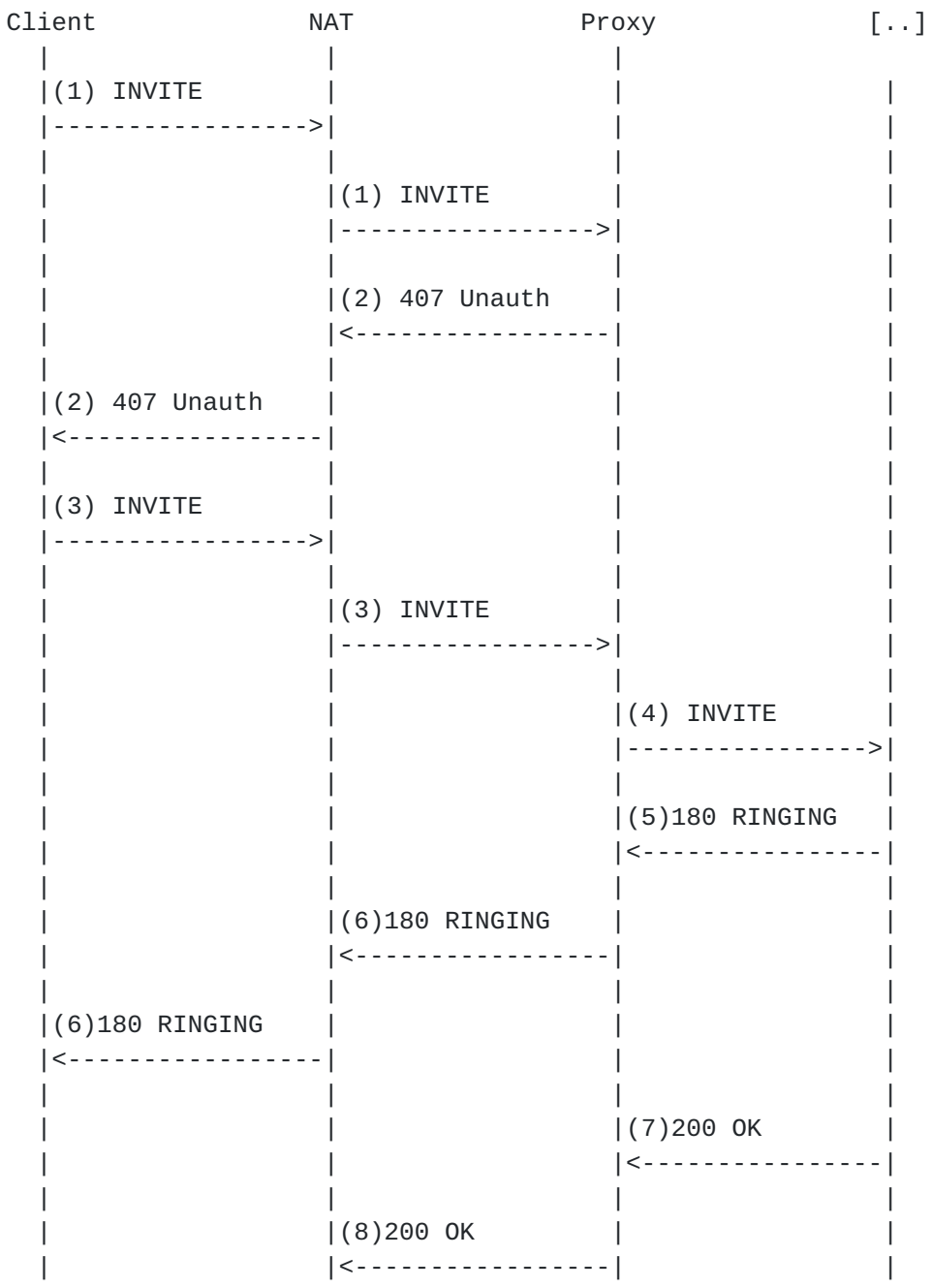
This REGISTER request results in the Path header being stored along with the AOR and it's associated binding at the Registrar. The URI contained in the Path header will be inserted as a pre-loaded SIP 'Route' header into any request that arrives at the Registrar and is directed towards the associated binding. This guarantees that all requests for the new Registration will be forwarded to the Edge Proxy. In our example, the user part of the SIP 'Path' header URI that was inserted by the Edge Proxy contains the unique token identifying the flow to the client. On receiving subsequent requests, the edge proxy will examine the user part of the pre-loaded SIP 'route' header and extract the unique flow token for use in its connection tuple comparison, as defined in the SIP Outbound specification [13]. An example which builds 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.



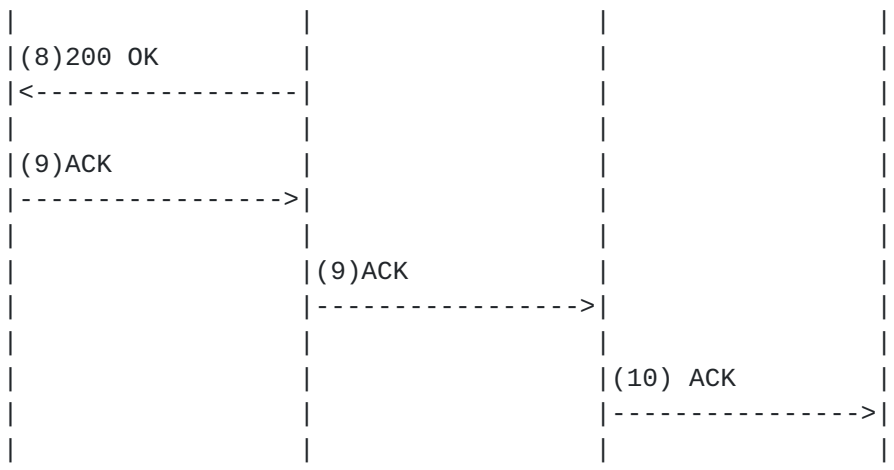


Figure 8.

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:clientA@example.com;gruu
   ;opaque=urn:uuid:ijed7ush-4jan-53120-aee5-e0aecwee6wef;grid=45a>
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, responses 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 [6].
2. Secondly, the contact inserted contains the GRUU previously obtained from the SIP 200 OK response to the registration. Use of the GRUU ensures that any SIP requests within the dialog that in the opposite direction will be able to traverse the NAT. This occurs using the mechanisms defined in the SIP Outbound

specification[13]. A request arriving at the entity which resolves to the GRUU is then able to determine a previously registered connection that will allow the request to traverse the NAT and reach the intended endpoint.

4.1.3.2. Reliable Transport

When using a reliable transport such as TCP the call flow and procedures for traversing a NAT are almost identical to those described in Section 4.1.3.1. The primary difference when using reliable transport protocols is that Symmetric response[6] are not required for SIP responses to traverse a NAT. RFC 3261[1] defines procedures for SIP response messages to be sent back on the same connection on which the request arrived.

4.1.4. Receiving an Invitation to a Session

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

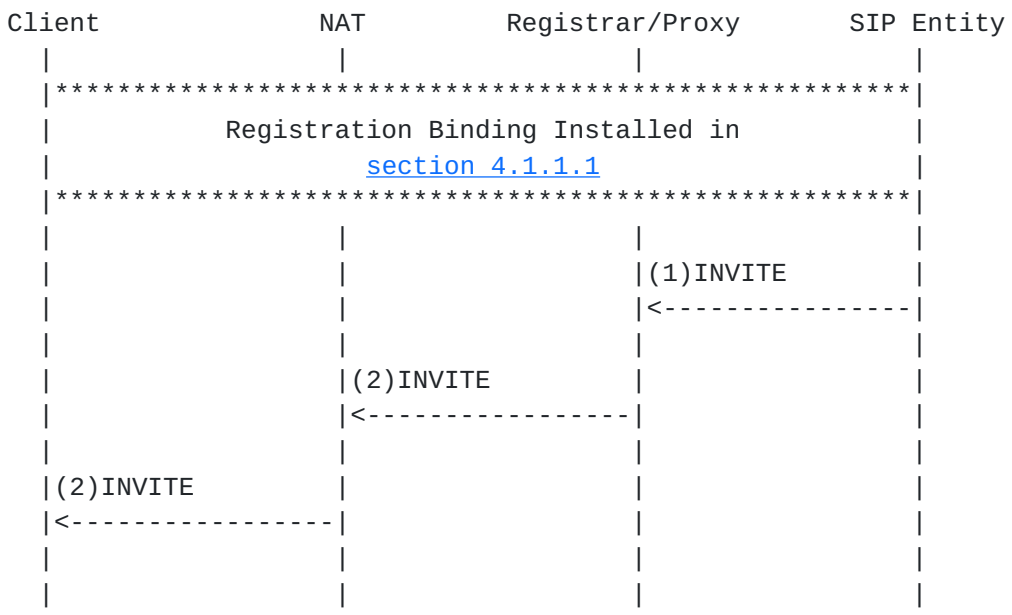


Figure 9.

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 request matches the registration binding previously installed 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 its list of current open flows. 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 flow. The request is then dispatched on the appropriate flow. 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 flow associated with the registration binding (look-up procedures in RFC 3263 [6] are overridden). 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

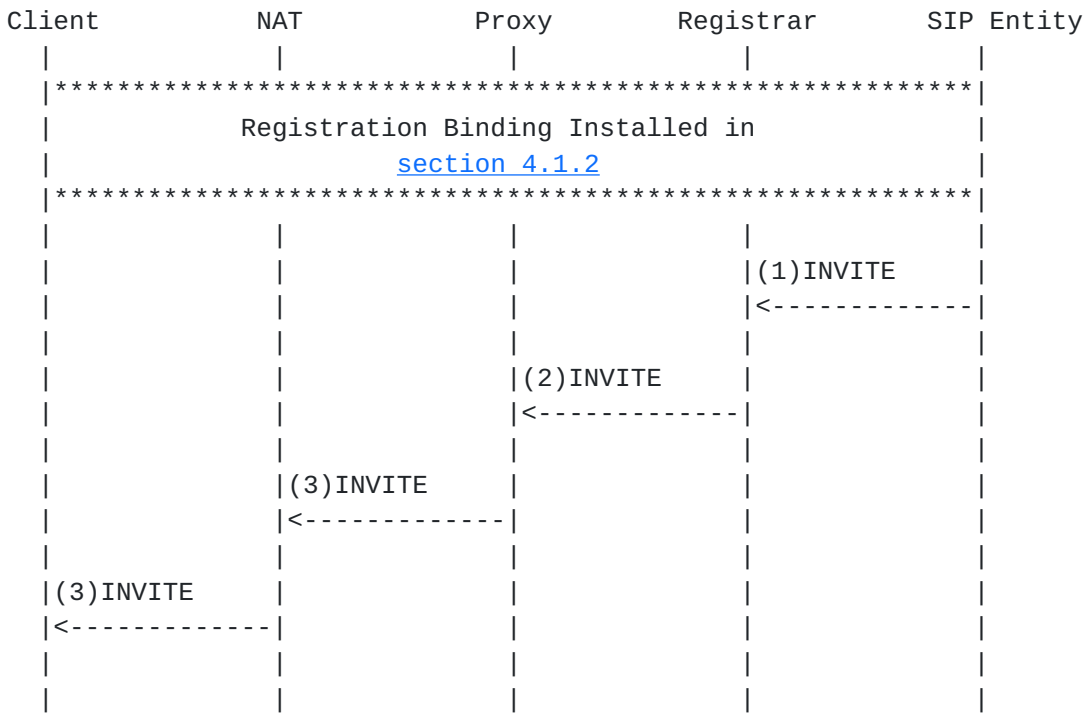


Figure 9.

4.2. Basic NAT Media Traversal

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

In the flow diagrams STUN messages have been annotated for simplicity as follows:

- The "Src" attribute represents the source transport address of the message.
- The "Dest" attribute represents the destination transport address of the message.
- The "Map" attribute represents the reflexive transport address.
- The "Rel" attribute represents the relayed transport address.

The meaning of each STUN attribute is extensively explained in the core STUN[10] and TURN usage[12] drafts.

The examples also contain a mechanism for representing transport addresses. It would be confusing to include representations of network addresses in the call flows and make them hard to follow. For this reason network addresses will be represented using the following annotation. The first component will contain the a representation of the client responsible for the address. For example in the majority of the examples "L" (left client), "R" (right

client), NAT-PUB" (NAT public), PRIV (Private), and "STUN-PUB" (STUN Public) are used. To allow for multiple addresses from the same network element, each representation can also be followed by a number. These can also be used in combination. For example "L-NAT-PUB-1" would represent a public network address on the left hand side NAT while "R-NAT-PUB-1" would represent a public network address in the right hand side of the NAT. "L-PRIV-1" would represent a private network address on the left hand side of the NAT while "R-PRIV-1" represents a private address on the right hand side of the NAT.

It should also be noted that during the examples it might be appropriate to signify an explicit part of a transport address. This is achieved by adding either the '.address' or '.port' tag on the end of the representation. For example, 'L-PRIV-1.address' and 'L-PRIV-1.port'.

4.2.1. Endpoint independent NAT

This section demonstrates an example of a client both initiating and receiving calls behind an 'Endpoint independent' NAT. An example is included for both STUN and ICE with ICE being the RECOMMENDED mechanism for media traversal.

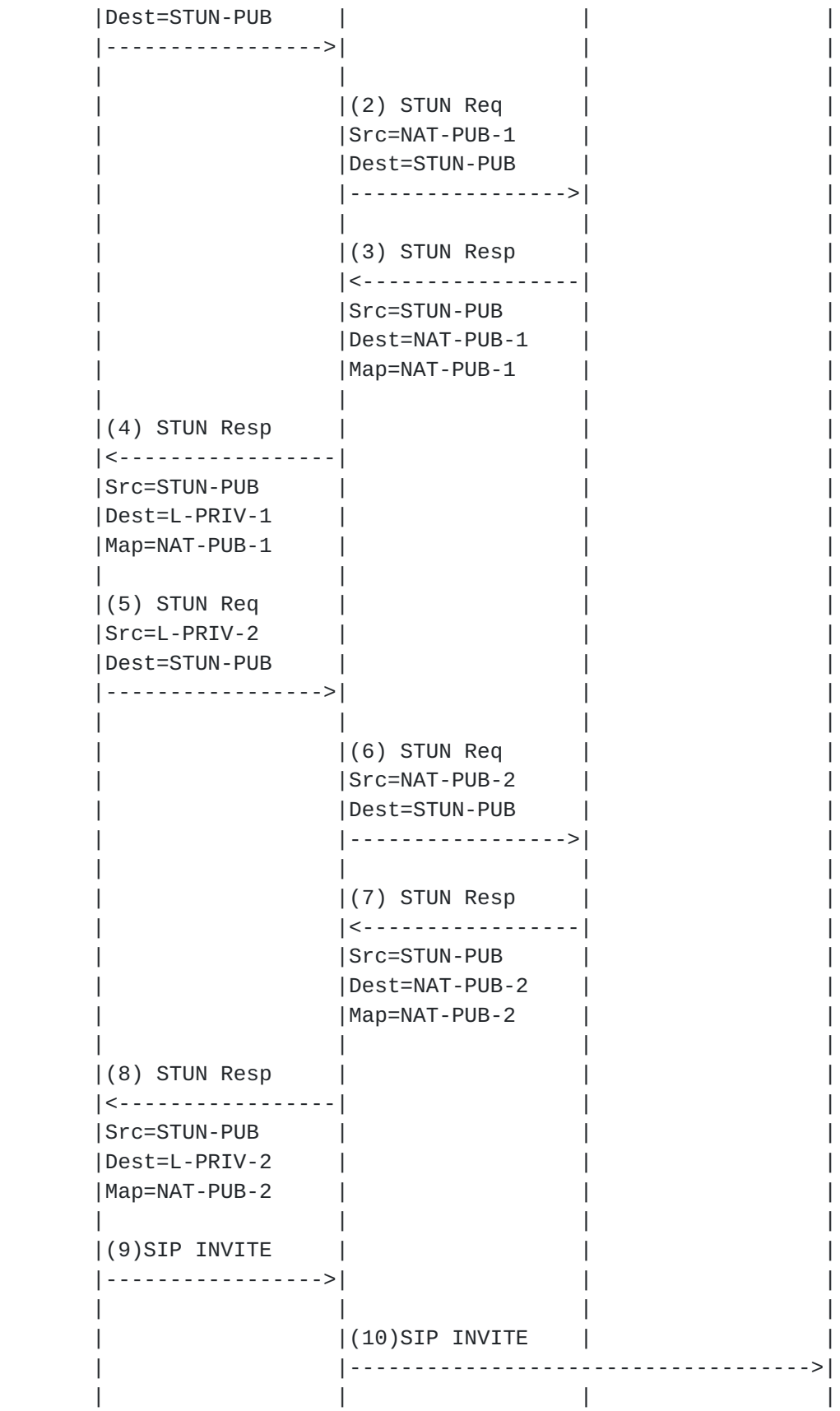
4.2.1.1. STUN Solution

It is possible to traverse media through an 'Endpoint Independent NAT using STUN. The remainder of this section provides a simplified examples of the 'Binding Discovery' STUN usage as defined in [10]. The STUN messages have been simplified and do not include 'Shared Secret' requests used to obtain the temporary username and password.

[Editors Note: Expand to show full flow in including Auth?.]

4.2.1.1.1. Initiating Session

The following example demonstrates media traversal through a NAT demonstrating 'Address Independent' NAT behavior 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.


|Dest=STUN-PUB |
|----------->|
|(2) STUN Req
|Src=NAT-PUB-1
|Dest=STUN-PUB
|----------->|
|(3) STUN Resp
|<-----------|
|Src=STUN-PUB
|Dest=NAT-PUB-1
|Map=NAT-PUB-1
|(4) STUN Resp
|<-----------|
|Src=STUN-PUB
|Dest=L-PRIV-1
|Map=NAT-PUB-1
|(5) STUN Req
|Src=L-PRIV-2
|Dest=STUN-PUB
|----------->|
|(6) STUN Req
|Src=NAT-PUB-2
|Dest=STUN-PUB
|----------->|
|(7) STUN Resp
|<-----------|
|Src=STUN-PUB
|Dest=NAT-PUB-2
|Map=NAT-PUB-2
|(8) STUN Resp
|<-----------|
|Src=STUN-PUB
|Dest=L-PRIV-2
|Map=NAT-PUB-2
|(9)SIP INVITE
|----------->|
|(10)SIP INVITE
|----------->|

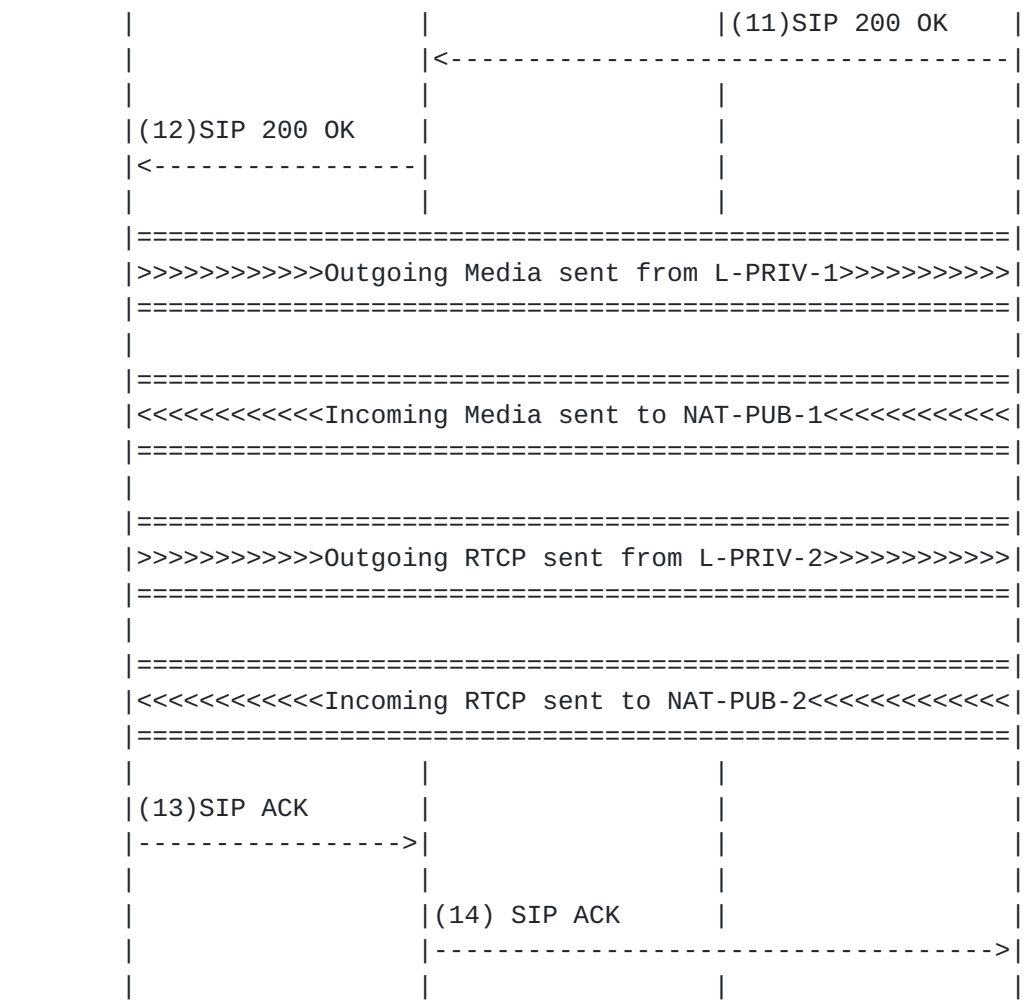


Figure 18: Address and Port Dependant NAT with STUN - Initiating

- On deciding to initiate a SIP voice session the client starts a local STUN client on the interface and port that is to be used for media (send/receive). 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 and STUN server.
- 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.
- 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=NAT-PUB-1' from (3)).

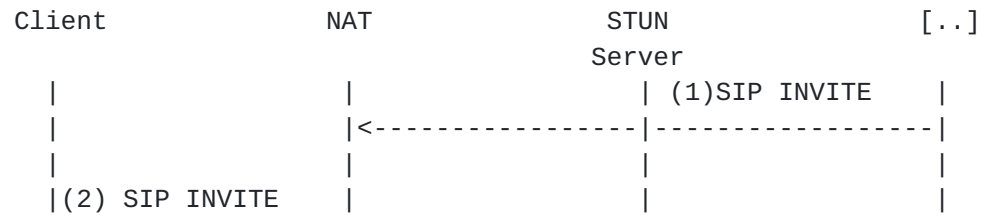
- The STUN response traverses back through the NAT using the binding created by the STUN request and presents the new mapped address to the client (4). At this point the process is repeated to obtain a second mapped address (as shown in (5)-(8)) for an alternative local address (Address has changed from "L-PRIV-1" to "L-PRIV-2").
- 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:

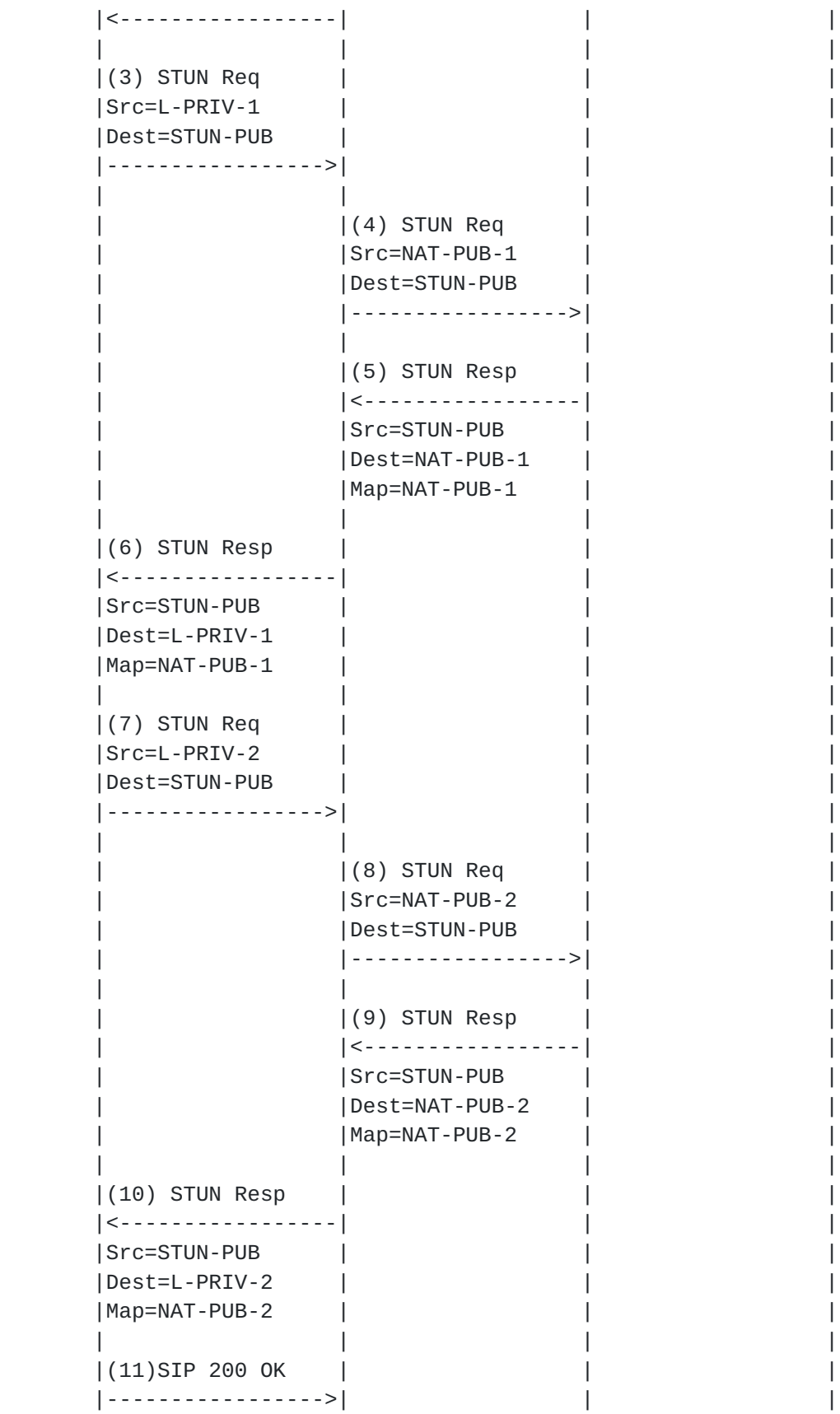
```
v=0
o=test 2890844526 2890842807 IN IP4 $L-PRIV-1.address
c=IN IP4 $NAT-PUB-1.address
t=0 0
m=audio $NAT-PUB-1.port RTP/AVP 0
a=rtcp:$NAT-PUB-2.port
```

- Note that the mapped address obtained from the STUN transactions are inserted as the connection address for the SDP (c=NAT-PUB-1.address). The Primary port for RTP is also inserted in the SDP (m=audio NAT-PUB-1.port RTP/AVP 0). Finally, the port gained from the additional STUN binding is placed in the RTCP attribute (as discussed in Section 3.2.1.1) for traversal of RTCP (a=rtcp:NAT-PUB-2.port).
- 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 and RTCP 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 an 'Address and Port dependant' 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.




(3) STUN Req
Src=L-PRIV-1
Dest=STUN-PUB
(4) STUN Req
Src=NAT-PUB-1
Dest=STUN-PUB
(5) STUN Resp
Src=STUN-PUB
Dest=NAT-PUB-1
Map=NAT-PUB-1
(6) STUN Resp
Src=STUN-PUB
Dest=L-PRIV-1
Map=NAT-PUB-1
(7) STUN Req
Src=L-PRIV-2
Dest=STUN-PUB
(8) STUN Req
Src=NAT-PUB-2
Dest=STUN-PUB
(9) STUN Resp
Src=STUN-PUB
Dest=NAT-PUB-2
Map=NAT-PUB-2
(10) STUN Resp
Src=STUN-PUB
Dest=L-PRIV-2
Map=NAT-PUB-2
(11)SIP 200 OK

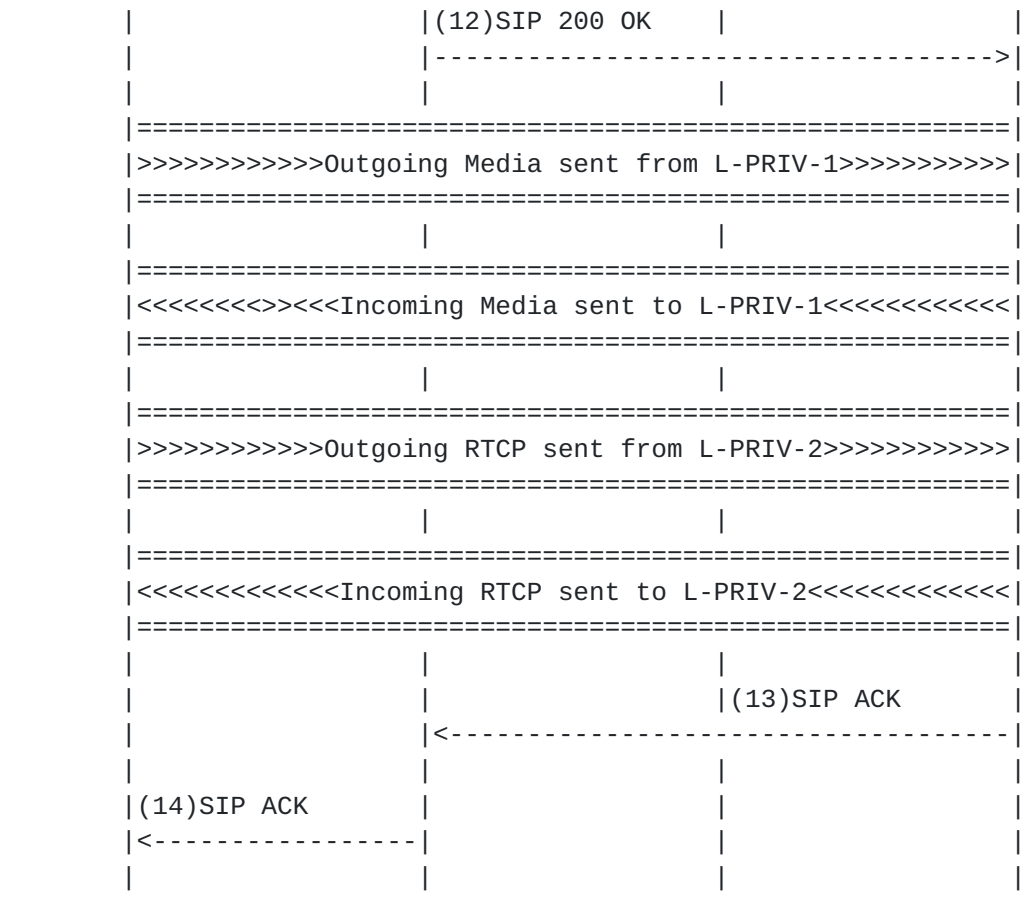


Figure 20: Restricted NAT with STUN - Receiving

- On receiving an invitation to a SIP voice session (SIP INVITE request) the User Agent starts a local STUN client on the appropriate port on which it is to receive media. 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 and STUN Server.
- 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.
- The STUN server receives the request and processes it 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=NAT-PUB-1' from (5)).

- The STUN response traverses back through the NAT using the binding created by the outgoing 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 and is represented by L-PRIV-2 in (7)).
- The client now constructs a SIP 200 OK message (11) in response to the original SIP INVITE requests. Note that traversal of SIP is not covered in this example and is discussed in earlier sections of the document. SIP Provisional responses are also left out for simplicity. The 200 OK response will use the addresses it has obtained in the previous STUN transactions to populate the SDP of the SIP 200 OK as shown below:

```
v=0
o=test 2890844526 2890842807 IN IP4 $L-PRIV-1.address
c=IN IP4 $NAT-PUB-1.address
t=0 0
m=audio $NAT-PUB-1.port RTP/AVP 0
a=rtcp:$NAT-PUB-2.port
```

- Note that the mapped address obtained from the initial STUN transaction is inserted as the connection address for the SDP (c=NAT-PUB-1.address). The Primary port for RTP is also inserted in the SDP (m=audio NAT-PUB-1.port RTP/AVP 0). Finally, the port gained from the additional binding is placed in the RTCP attribute (as discussed in Section 3.2.1.1) for traversal of RTCP (a=rtcp: NAT-PUB-2.port).
- 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.

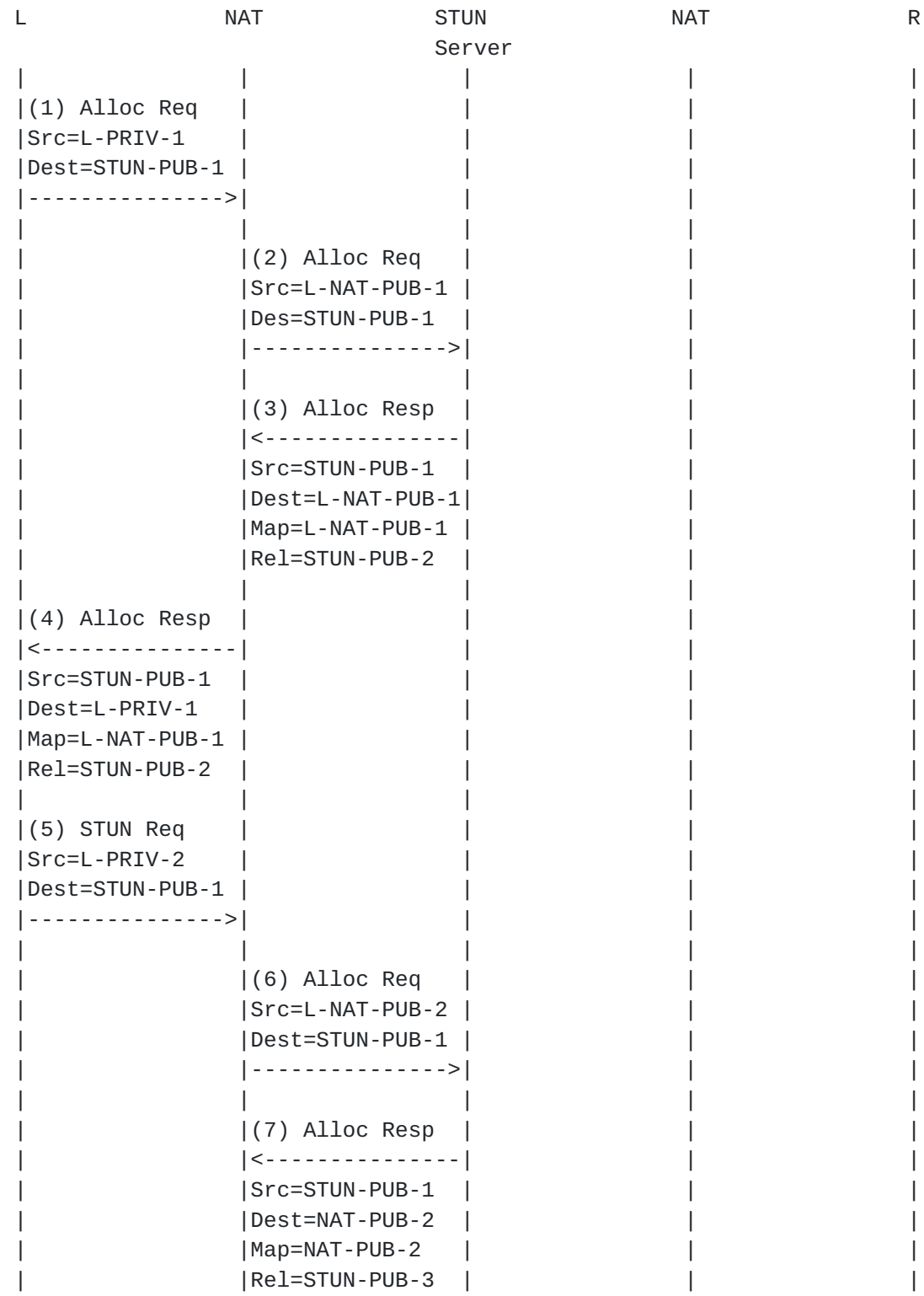
4.2.1.2. ICE Solution

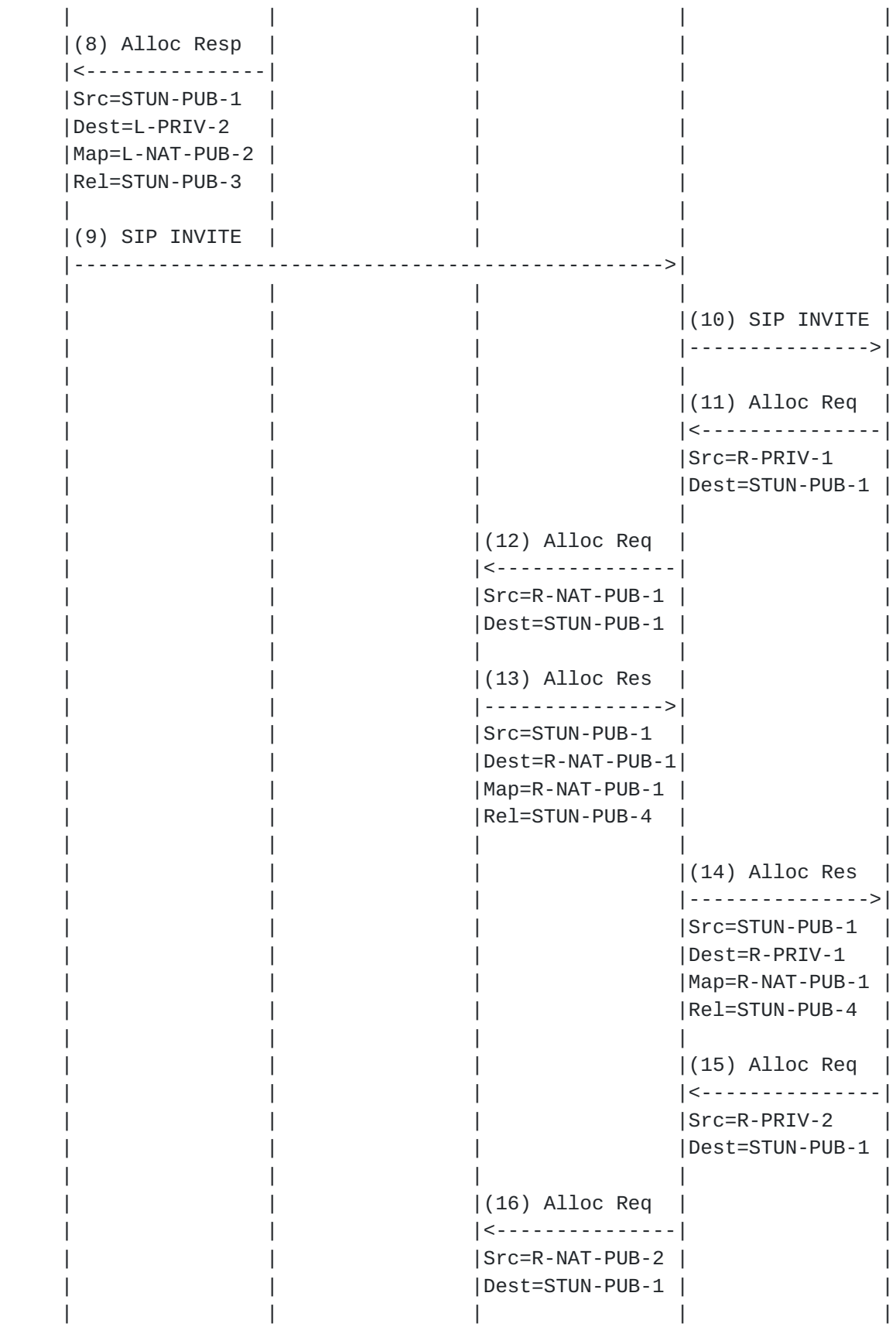
The preferred solution for media traversal of NAT is using ICE, as described in Section 3.2.4, regardless of the NAT type. The following examples illustrate the traversal of an 'Endpoint independent' NAT for both an initiating. The example only covers ICE in association with STUN and TURN usage.

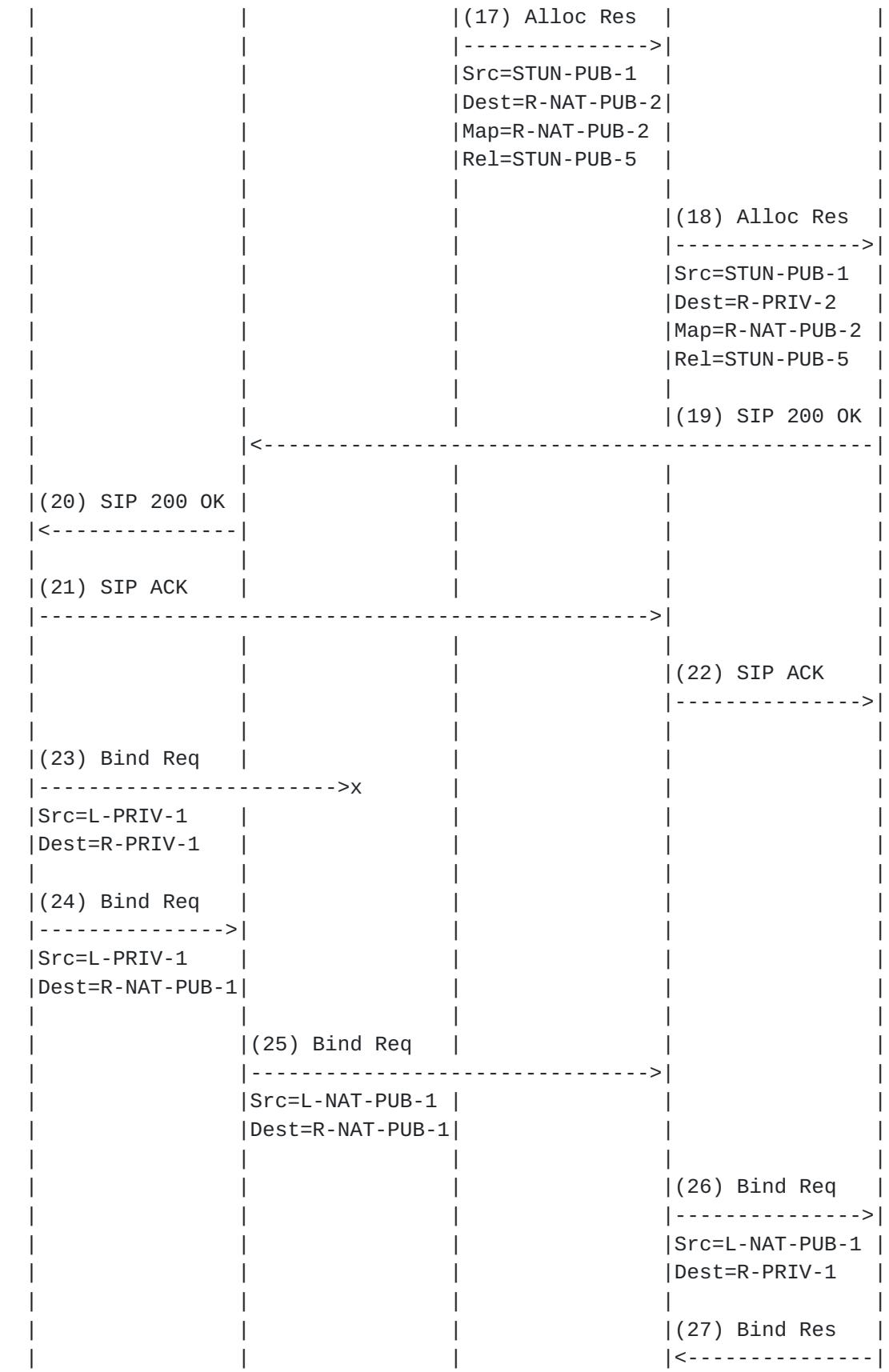
4.2.1.2.1. Initiating Session

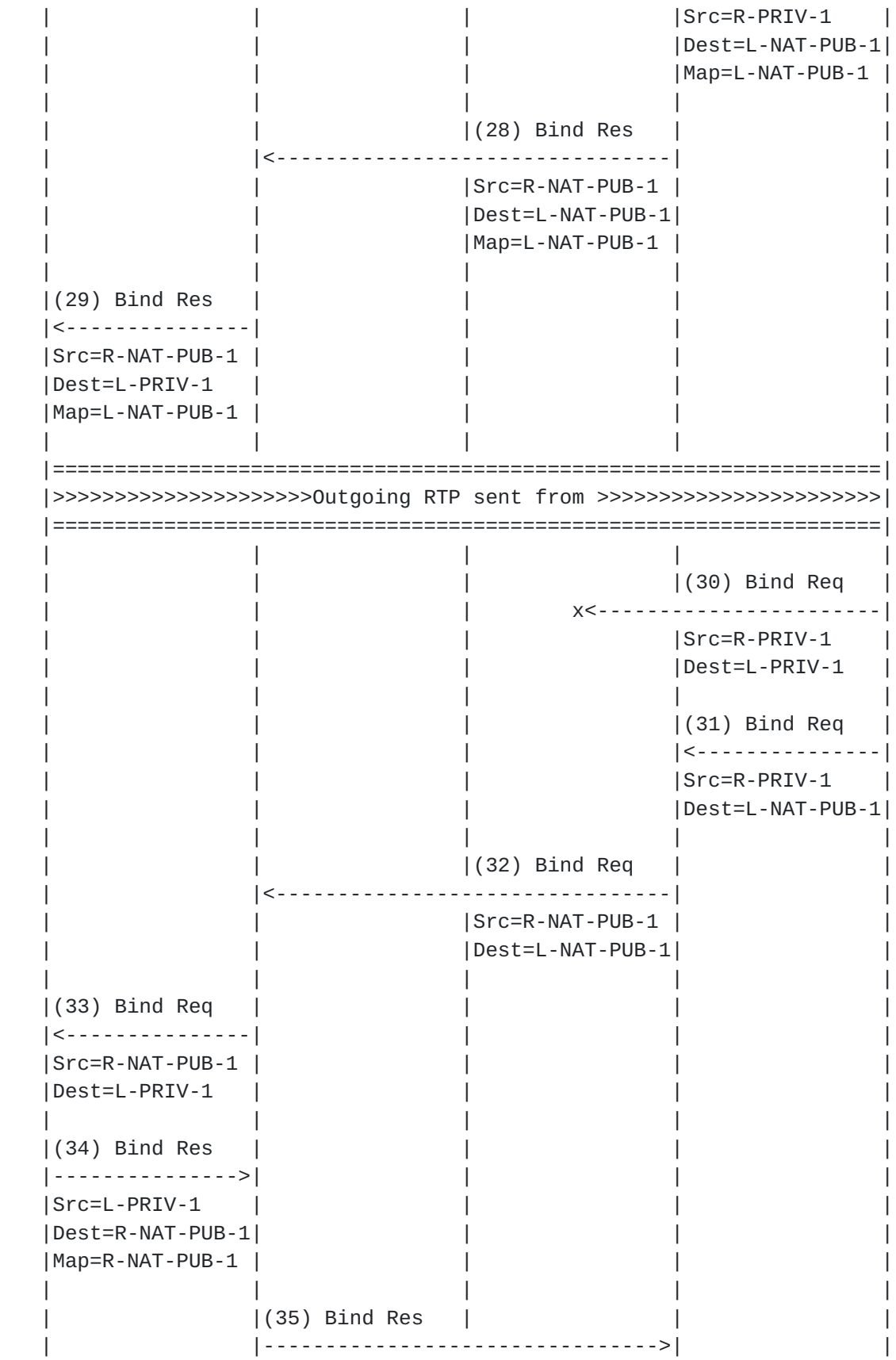
The following example demonstrates an initiating traversal through an 'Endpoint independent' NAT using ICE.

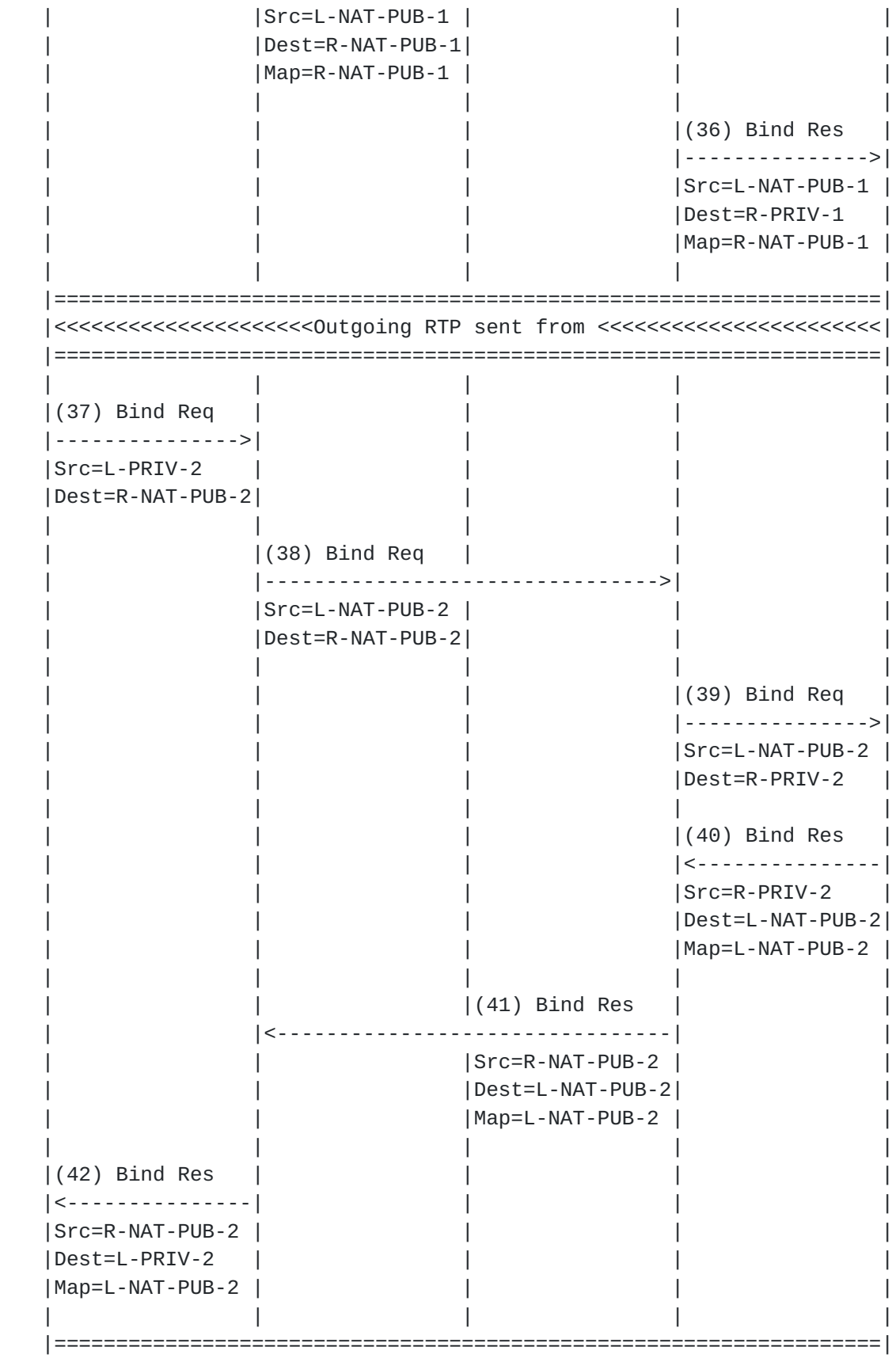
[Editors Note: Example needs to be expanded to include more ICE detail e.g. timers etc.]

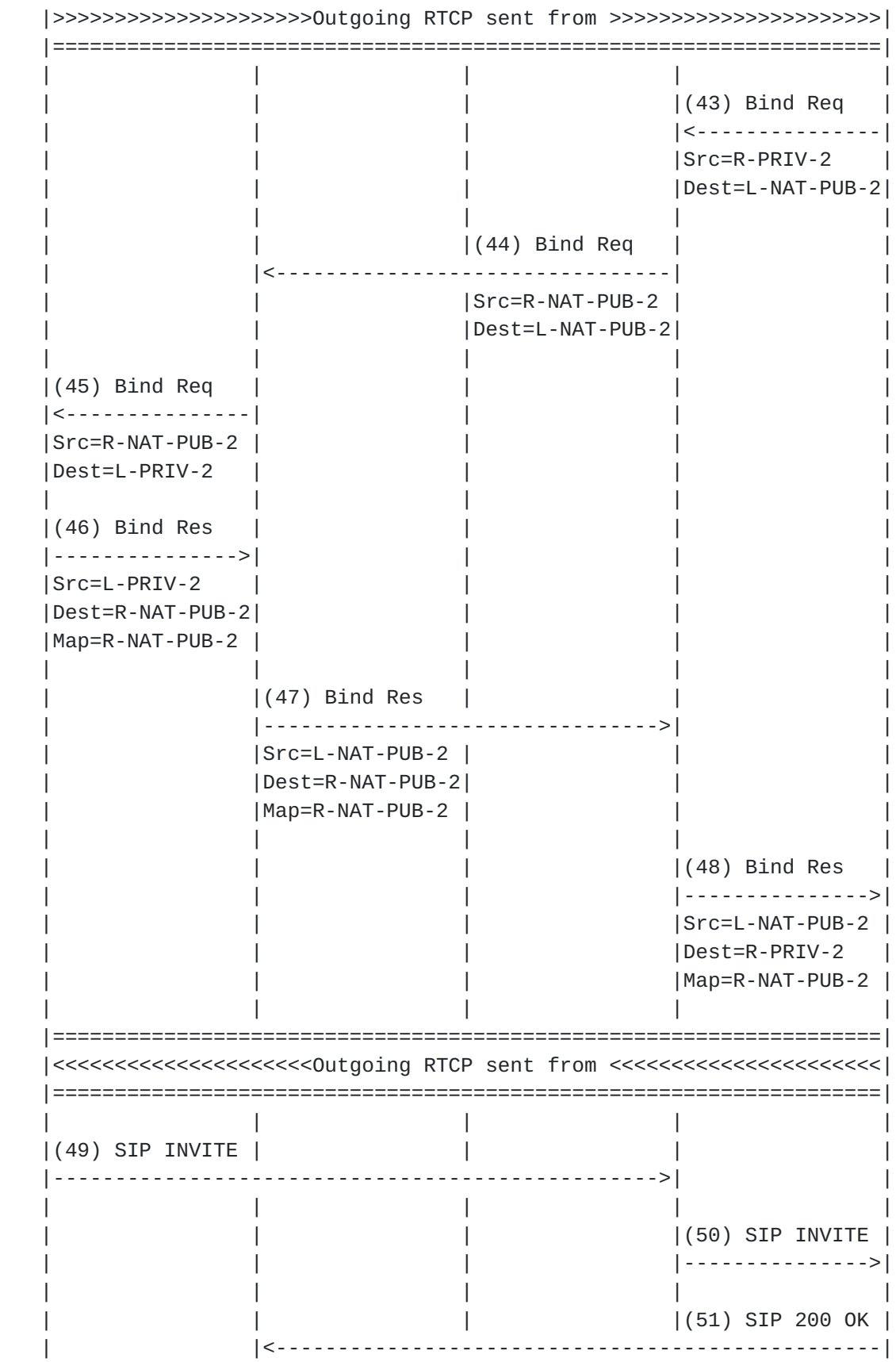



(8) Alloc Resp
Src=STUN-PUB-1
Dest=L-PRIV-2
Map=L-NAT-PUB-2
Rel=STUN-PUB-3
(9) SIP INVITE
(10) SIP INVITE
(11) Alloc Req
Src=R-PRIV-1
Dest=STUN-PUB-1
(12) Alloc Req
Src=R-NAT-PUB-1
Dest=STUN-PUB-1
(13) Alloc Res
Src=STUN-PUB-1
Dest=R-NAT-PUB-1
Map=R-NAT-PUB-1
Rel=STUN-PUB-4
(14) Alloc Res
Src=STUN-PUB-1
Dest=R-PRIV-1
Map=R-NAT-PUB-1
Rel=STUN-PUB-4
(15) Alloc Req
Src=R-PRIV-2
Dest=STUN-PUB-1
(16) Alloc Req
Src=R-NAT-PUB-2
Dest=STUN-PUB-1


(17) Alloc Res
Src=STUN-PUB-1
Dest=R-NAT-PUB-2
Map=R-NAT-PUB-2
Rel=STUN-PUB-5
(18) Alloc Res
Src=STUN-PUB-1
Dest=R-PRIV-2
Map=R-NAT-PUB-2
Rel=STUN-PUB-5
(19) SIP 200 OK
(20) SIP 200 OK
(21) SIP ACK
(22) SIP ACK
(23) Bind Req
Src=L-PRIV-1
Dest=R-PRIV-1
(24) Bind Req
Src=L-PRIV-1
Dest=R-NAT-PUB-1
(25) Bind Req
Src=L-NAT-PUB-1
Dest=R-NAT-PUB-1
(26) Bind Req
Src=L-NAT-PUB-1
Dest=R-PRIV-1
(27) Bind Res


Src=R-PRIV-1
Dest=L-NAT-PUB-1
Map=L-NAT-PUB-1
(28) Bind Res
Src=R-NAT-PUB-1
Dest=L-NAT-PUB-1
Map=L-NAT-PUB-1
(29) Bind Res
Src=R-NAT-PUB-1
Dest=L-PRIV-1
Map=L-NAT-PUB-1
>>>>>>>>>>>>>>>>>>>Outgoing RTP sent from >>>>>>>>>>>>>>>>>>>>
(30) Bind Req
Src=R-PRIV-1
Dest=L-PRIV-1
(31) Bind Req
Src=R-PRIV-1
Dest=L-NAT-PUB-1
(32) Bind Req
Src=R-NAT-PUB-1
Dest=L-NAT-PUB-1
(33) Bind Req
Src=R-NAT-PUB-1
Dest=L-PRIV-1
(34) Bind Res
Src=L-PRIV-1
Dest=R-NAT-PUB-1
Map=R-NAT-PUB-1
(35) Bind Res


Src=L-NAT-PUB-1
Dest=R-NAT-PUB-1
Map=R-NAT-PUB-1
(36) Bind Res
Src=L-NAT-PUB-1
Dest=R-PRIV-1
Map=R-NAT-PUB-1
<<<<<<<<<<<<<<<<<<<<Outgoing RTP sent from <<<<<<<<<<<<<<<<<<<<
(37) Bind Req
Src=L-PRIV-2
Dest=R-NAT-PUB-2
(38) Bind Req
Src=L-NAT-PUB-2
Dest=R-NAT-PUB-2
(39) Bind Req
Src=L-NAT-PUB-2
Dest=R-PRIV-2
(40) Bind Res
Src=R-PRIV-2
Dest=L-NAT-PUB-2
Map=L-NAT-PUB-2
(41) Bind Res
Src=R-NAT-PUB-2
Dest=L-NAT-PUB-2
Map=L-NAT-PUB-2
(42) Bind Res
Src=R-NAT-PUB-2
Dest=L-PRIV-2
Map=L-NAT-PUB-2


>>>>>>>>>>>>>>>>>>>Outgoing RTCP sent from >>>>>>>>>>>>>>>>>>>>
(43) Bind Req
Src=R-PRIV-2
Dest=L-NAT-PUB-2
(44) Bind Req
Src=R-NAT-PUB-2
Dest=L-NAT-PUB-2
(45) Bind Req
Src=R-NAT-PUB-2
Dest=L-PRIV-2
(46) Bind Res
Src=L-PRIV-2
Dest=R-NAT-PUB-2
Map=R-NAT-PUB-2
(47) Bind Res
Src=L-NAT-PUB-2
Dest=R-NAT-PUB-2
Map=R-NAT-PUB-2
(48) Bind Res
Src=L-NAT-PUB-2
Dest=R-PRIV-2
Map=R-NAT-PUB-2
<<<<<<<<<<<<<<<<<<<<Outgoing RTCP sent from <<<<<<<<<<<<<<<<<<<<
(49) SIP INVITE
(50) SIP INVITE
(51) SIP 200 OK

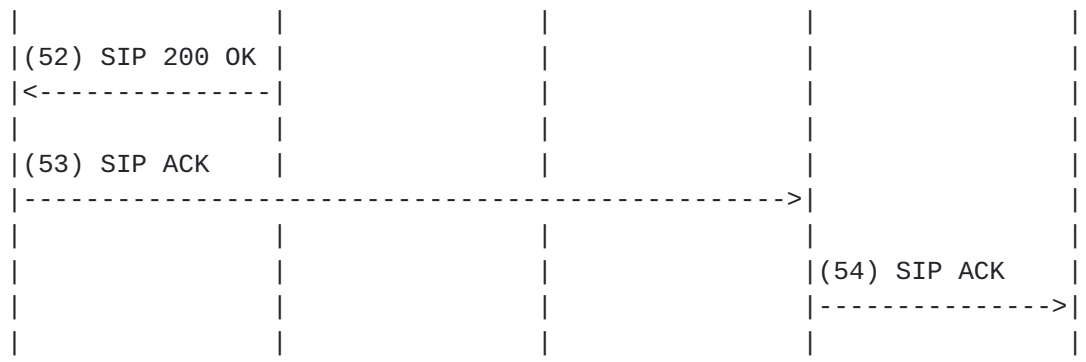


Figure 22: Endpoint Independent NAT with ICE

- On deciding to initiate a SIP voice session the SIP client 'L' starts a local STUN client. The STUN client generates a standard Allocate request as indicated in (1) from Figure 22 which also highlights the source address and port combination for which the client device wishes to obtain a mapping. The STUN request is sent through the NAT towards the public internet.
- Allocate message (2) traverses the NAT and breaks out onto the public internet towards the public STUN server. Note that the source address of the Allocate request now represents the public address and port from the public side of the NAT (L-NAT-PUB-1).
- The STUN server receives the Allocate request and processes appropriately. This results in a successful Allocate response being generated and returned (3). The message contains details of the reflexive address which is to be used by the originating client to receive media (see 'Map=L-NAT-PUB-1') from (3)). It also contains an appropriate relay address that can be used at the STUN server (see 'Rel=STUN-PUB-2').
- The STUN response traverses back through the NAT using the binding created by the initial Allocate request and presents the new mapped address to the client (4). The process is repeated and a second STUN derived set of address' are obtained, as illustrated in (5)-(8) in Figure 22. At this point the User Agent behind the NAT has pairs of derived external reflexive and relayed representations. The client would be free to gather any number of external representations using any UNSAF[9] compliant protocol.
- The client now constructs a SIP INVITE message (9). 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[16], as shown below in Figure 23 (*note - /* signifies line continuation):

```
v=0
o=test 2890844526 2890842807 IN IP4 $L-PRIV-1
c=IN IP4 $L-PRIV-1.address
t=0 0
a=ice-pwd:$LPASS
m=audio $L-PRIV-1.port RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=rtcp:$L-PRIV-2.port
a=candidate:$L1 1 UDP 1.0 $L-PRIV-1.address $L-PRIV-1.port
a=candidate:$L1 2 UDP 1.0 $L-PRIV-2.address $L-PRIV-2.port
a=candidate:$L2 1 UDP 0.7 $L-NAT-PUB-1.address $L-NAT-PUB-1.port
a=candidate:$L2 2 UDP 0.7 $L-NAT-PUB-2.address $L-NAT-PUB-2.port
a=candidate:$L3 1 UDP 0.3 $STUN-PUB-2.address $STUN-PUB-2.port
a=candidate:$L3 2 UDP 0.3 $STUN-PUB-3.address $STUN-PUB-3.port
```

Figure 23: ICE SDP Offer

- The SDP has been constructed to include all the available candidate pairs that have been assembled. The first candidate pair (as identified by $L1) contain two local addresses that have the highest priority (1.0). They are also encoded into the connection (c=) and media (m=) lines of the SDP. The second 'candidate' address pair, as identified by the component-id, contains the two NAT reflexive addresses obtained from the STUN server for both RTP and RTCP traffic (identified by candidate-id $L2). This entry has been given a priority (0.7) by the client. The third and final candidate pair represents the relayed addresses (as identified by $L3) obtained from the STUN server. This pair has the lowest priority (0.3) and will be used as a last resort.
- The SIP signalling then traverses the NAT and sets up the SIP session (9)-(10). 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.
- On receiving the SIP INVITE request (10) client 'R' also starts local STUN servers on appropriate address/port combinations and gathers potential candidate addresses to be encoded into the SDP. Steps (11-18) involve client 'R' carrying out the same steps as client 'L'. This involves obtaining local, reflexive and relayed addresses. Client 'R' is now ready to generate an appropriate answer in the SIP 200 OK message (19). The example answer follows in Figure 23 (*note - /* signifies line continuation):

```
v=0
o=test 3890844516 3890842803 IN IP4 $R-PRIV-1
c=IN IP4 $R-PRIV-1.address
t=0 0
a=ice-pwd:$RPASS
m=audio $R-PRIV-1.port RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=rtcp:$R-PRIV-2.port
a=candidate:$L1 1 UDP 1.0 $R-PRIV-1.address $R-PRIV-1.port
a=candidate:$L1 2 UDP 1.0 $R-PRIV-2.address $R-PRIV-2.port
a=candidate:$L2 1 UDP 0.7 $R-NAT-PUB-1.address $R-NAT-PUB-1.port
a=candidate:$L2 2 UDP 0.7 $R-NAT-PUB-2.address $R-NAT-PUB-2.port
a=candidate:$L3 1 UDP 0.3 $STUN-PUB-2.address $STUN-PUB-4.port
a=candidate:$L3 2 UDP 0.3 $STUN-PUB-3.address $STUN-PUB-5.port
```

Figure 24: ICE SDP Answer

- The two clients will now form candidate pairs and the transport address check list as specified in ICE. Both 'L' and 'R' will start the check list with the currently active component pair (contained in the 'c=' and 'm=' of the SDP). As illustrated in (23), client 'L' constructs a STUN Bind request in an attempt to validate the connection address received in the SDP of the 200 OK (20). As a private address was specified in the active address in the SDP, the Stun Bind request fails to reach its destination causing a bind failure. Client 'L' now attempts a STUN Bind request for the first candidate pair in the returned SDP with the highest priority (24). As can be seen from messages (24) to (29), the STUN Bind request is successful and returns a positive outcome for the connectivity check. Client 'L' is now free to steam media to the candidate pair. Client 'R' also carries out the same set of binding requests. It firstly (in parallel) tries to contact the active address contained in the SDP (30). Client 'R' now attempts a STUN Bind request for the first candidate pair in the returned SDP with the highest priority (31). As can be seen from messages (31) to (36), the STUN bind request is successful and returns a positive outcome for the connectivity check. The previously described check confirm on both sides that connectivity can be achieved through appropriate candidates. As part of the process in this example, both 'L' and 'R' will now complete the same connectivity checks for part 2 of the component named for each candidate for use with RTCP. The connectivity checks for part '2' of the candidate component are shown in 'L'(37-42) and 'R'(43-48).
- The candidates have now been fully verified (Valid status) and as they are the highest priority, an updated offer (49-50) is now sent from the offerer (client 'L') to the answerer (client 'R'

representing the new active candidates. The new offer would look as follows:

```
v=0
o=test 2890844526 2890842808 IN IP4 $L-PRIV-1
c=IN IP4 $L-NAT-PUB-1.address
t=0 0
a=ice-pwd:$LPASS
m=audio $L-NAT-PUB-1.port RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=rtcp:$L-NAT-PUB-2.port
a=candidate:$L2 1 UDP 0.7 $L-NAT-PUB-1.address $L-NAT-PUB-1.port
a=candidate:$L2 2 UDP 0.7 $L-NAT-PUB-2.address $L-NAT-PUB-2.port
```

Figure 25: ICE SDP Updated Offer

- The resulting answer (51-52) for 'R' would look as follows:

```
v=0
o=test 3890844516 3890842803 IN IP4 $R-PRIV-1
c=IN IP4 $R-PRIV-1.address
t=0 0
a=ice-pwd:$RPASS
m=audio $R-PRIV-1.port RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=rtcp:$R-PRIV-2.port
a=candidate:$L2 1 UDP 0.7 $R-NAT-PUB-1.address $R-NAT-PUB-1.port
a=candidate:$L2 2 UDP 0.7 $R-NAT-PUB-2.address $R-NAT-PUB-2.port
```

Figure 26: ICE SDP Updated Answer

4.2.2. Port and Address Dependant 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 case. The use of basic STUN on its own will not guarantee traversal through every NAT type, hence the recommendation that ICE is the preferred option.

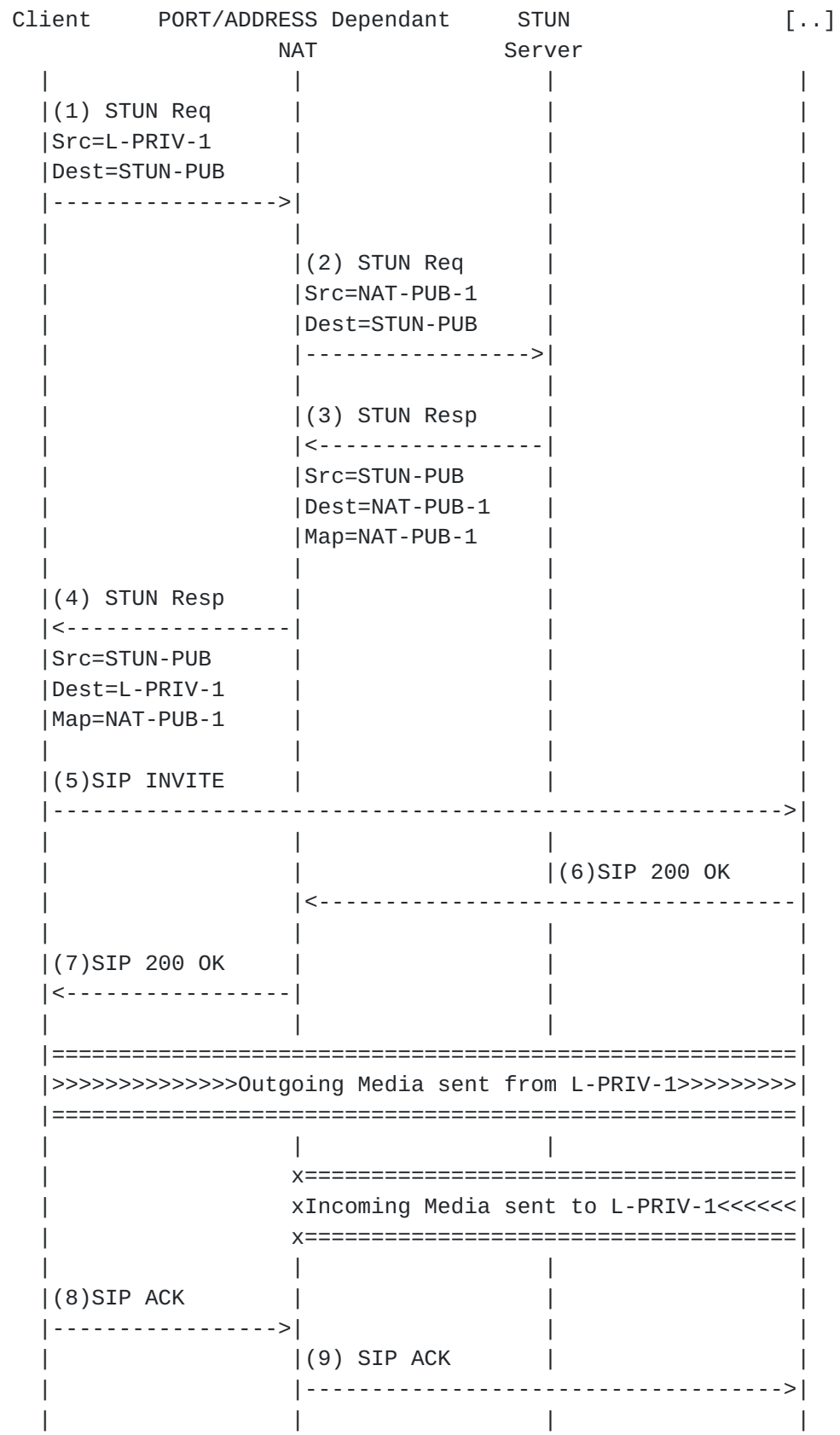

Client
PORT/ADDRESS Dependant NAT
STUN Server
[..]
(1) STUN Req
Src=L-PRIV-1
Dest=STUN-PUB
(2) STUN Req
Src=NAT-PUB-1
Dest=STUN-PUB
(3) STUN Resp
Src=STUN-PUB
Dest=NAT-PUB-1
Map=NAT-PUB-1
(4) STUN Resp
Src=STUN-PUB
Dest=L-PRIV-1
Map=NAT-PUB-1
(5)SIP INVITE
(6)SIP 200 OK
(7)SIP 200 OK
>>>>>>>>>>>>>Outgoing Media sent from L-PRIV-1>>>>>>>>>
xIncoming Media sent to L-PRIV-1<<<<<<
(8)SIP ACK
(9) SIP ACK

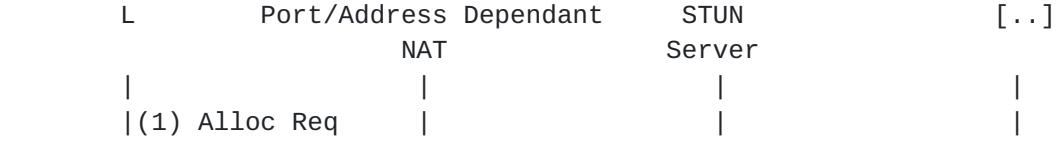
Figure 27: Port/Address Dependant NAT with STUN - Failure

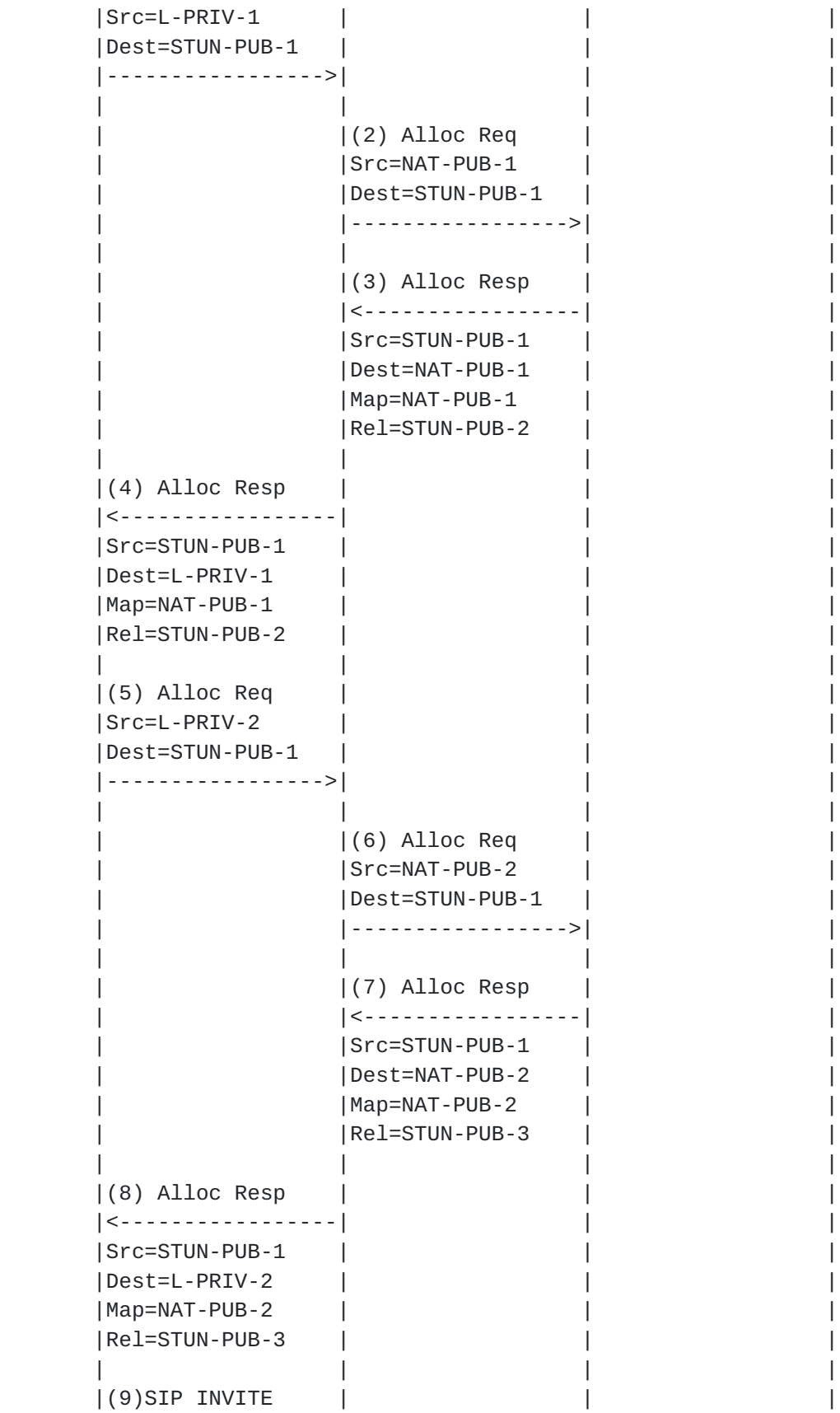
The example in Figure 27 is conveyed in the context of a client behind the 'Port/Address Dependant' NAT initiating a call. It should be noted that the same problem applies when a client receives a SIP invitation and is behind a Port/Address Dependant NAT.

- o In Figure 27 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 (reflexive) 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 reflexive address supplied in the outgoing INVITE request to traverse media back through the 'Port/Address Dependent' NAT.
- o A 'Port/Address Dependant' NAT has differing rules from the 'Endpoint Independent' type of NAT (as defined in [11]). For any internal IP address and port combination, data sent to a different external destination does not provide the same public mapping at the NAT. In Figure 27 the STUN query produced a valid external mapping or receiving media. 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 optionally port, will be dropped at the NAT. Figure 27 shows the media being dropped at the NAT after (7) and before (8). This then leads to one way audio.

4.2.2.2. TURN Usage Solution

As identified in Section 4.2.2.1, STUN provides a useful tool kit for the traversal of the majority of NATs but fails with Port/Address Dependant NAT. This led to the development of the TURN usage solution [12] which uses the STUN toolkit. It allows a client to request a relayed address at the STUN server rather than a reflexive representation. This then introduces a media relay in the path for NAT traversal (as described in Section 3.2.3). The following example explains how the TURN usage solves the previous failure when using STUN to traverse a 'Port/Address Dependant' type NAT.




Src=L-PRIV-1		
Dest=STUN-PUB-1		
----------->		
	(2) Alloc Req	
	Src=NAT-PUB-1	
	Dest=STUN-PUB-1	
	----------->	
	(3) Alloc Resp	
	<-----------	
	Src=STUN-PUB-1	
	Dest=NAT-PUB-1	
	Map=NAT-PUB-1	
	Rel=STUN-PUB-2	
(4) Alloc Resp		
<-----------		
Src=STUN-PUB-1		
Dest=L-PRIV-1		
Map=NAT-PUB-1		
Rel=STUN-PUB-2		
(5) Alloc Req		
Src=L-PRIV-2		
Dest=STUN-PUB-1		
----------->		
	(6) Alloc Req	
	Src=NAT-PUB-2	
	Dest=STUN-PUB-1	
	----------->	
	(7) Alloc Resp	
	<-----------	
	Src=STUN-PUB-1	
	Dest=NAT-PUB-2	
	Map=NAT-PUB-2	
	Rel=STUN-PUB-3	
(8) Alloc Resp		
<-----------		
Src=STUN-PUB-1		
Dest=L-PRIV-2		
Map=NAT-PUB-2		
Rel=STUN-PUB-3		
(9)SIP INVITE		

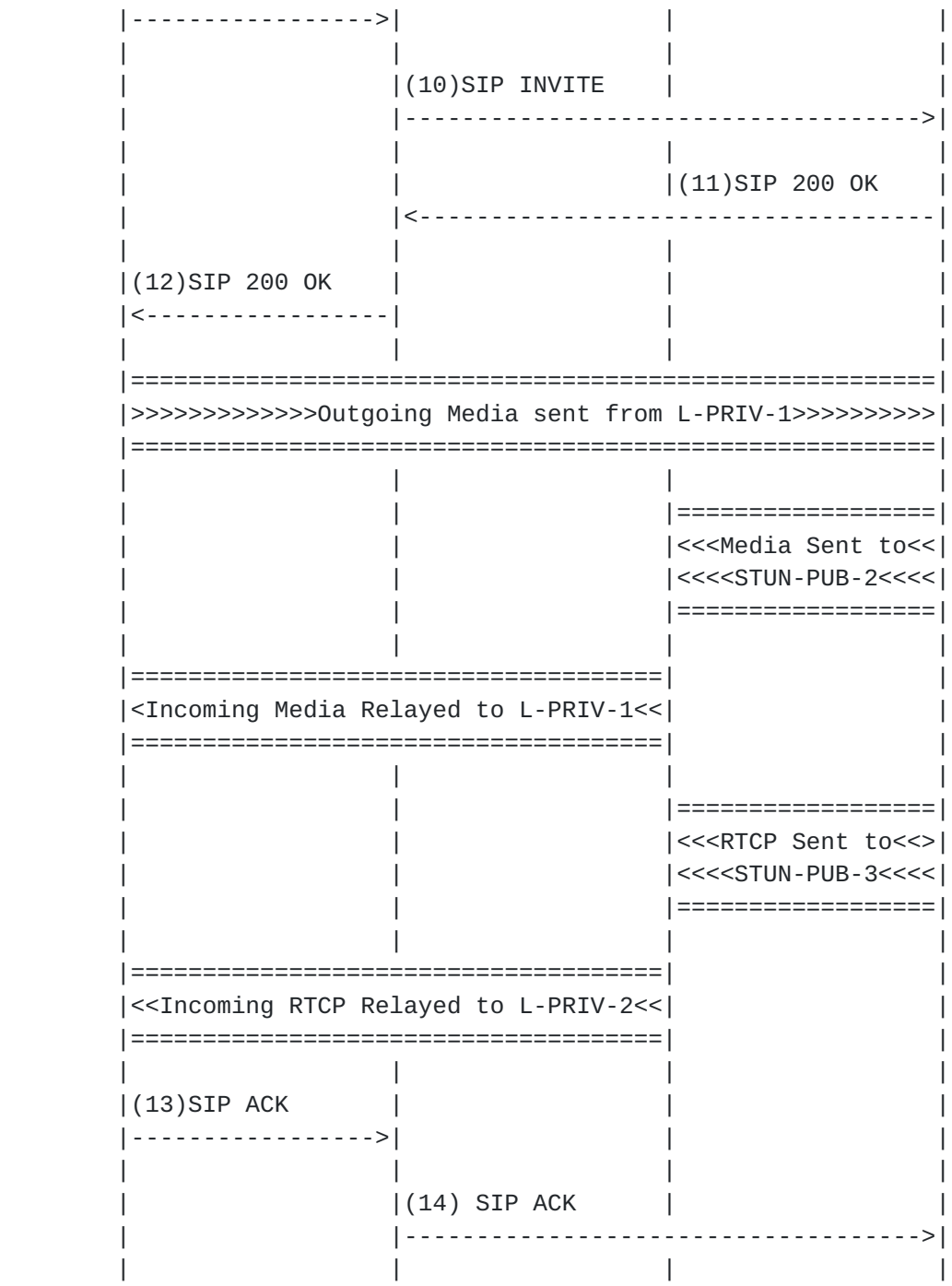


Figure 28: Port/Address Dependant NAT with TURN Usage - Success

o In this example, client 'L' issues a TURN allocate request(1) to obtained a relay address at the STUN server. The request traverses through the 'Port/Address Dependant' NAT and reaches the STUN server (2). The STUN server generates an Allocate response (3) that contains both a reflexive address (Map=NAT-PUB-1) of the

client and also a relayed address (Rel=STUN-PUB-2). The relayed address maps to an address mapping on the STUN server which is bound to the public pin hole that has been opened on the NAT by the Allocate request. This results in any traffic sent to the STUN server relayed address (Rel=STUN-PUB-2) being forwarded to the external representation of the pin hole created by the Allocate request(NAT-PUB-1).

- The TURN derived address (STUN-PUB-2) arrives back at the originating client(4) in an Allocate response. This address can then be used in the SDP for the outgoing SIP INVITE request as shown in the following example (note that the example also includes client 'L' obtaining a second relay address for use in the RTCP attribute (5-8)):
-

```
v=0
o=test 2890844342 2890842164 IN IP4 $L-PRIV-1
c=IN IP4 $STUN-PUB-2.address
t=0 0
m=audio $STUN-PUB-2.port RTP/AVP 0
a=rtcp:$STUN-PUB-3.port
```

- On receiving the INVITE request, the UAS is able to stream media and RTCP to the relay address (STUN-PUB-2 and STUN-PUB-3) at the STUN server. As shown in Figure 28 (between messages (12) and (13), the media from the UAS is directed to the relayed address at the STUN server. The STUN server then forwards the traffic to the open pin holes in the Port/Address Dependant NAT (NAT-PUB-1 and NAT-PUB-2). The media traffic is then able to traverse the 'Port/Address Dependant' NAT and arrives back at client 'L'.
- The TURN usage of STUN on its own will work for 'Port/Address Dependent' 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 NAT traversal and comes at a high processing cost.

4.2.2.3. ICE Solution

The previous two examples have highlighted the problem with using core STUN usage for all forms of NAT traversal and a solution using TURN usage for the Port/Address Dependant 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 makes use of core STUN, TURN usage and any other UNSAF[9] 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. These examples are associated with an 'Endpoint

Independent' type NAT but can be applied to any NAT type variation, including 'Port/Address Dependant' 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 specification[16]

[Editors Note: TODO - a detailed example will be included here which includes promotion of a TURN relayed address to the active candidate to traverse a 'Port/Address Dependent' type NAT.]

4.3. Address independent Port Restricted NAT --> Address independent Port Restricted NAT traversal

[Editors Note: TODO - a detailed example will be included where User A and B are both behind Address Independent NATs that have Port restricted properties. This means that the stun-derived addresses will work, but each side must send a 'suicide' or 'primer' STUN packet that creates a permission in the NAT. So, the main thing to show here is how the first packet from B to A will create a permission in B's NAT but gets dropped at A. When A gets the answer it starts its STUN checks and the packet from A to B creates a permission in A's NAT and gets through B's NAT because of the previously installed permission. This now triggers B to resend its stun request which now works.]

4.4. Internal TURN Usage (Enterprise Deployment)

[Editors Note: TODO - a detailed example will be included for User A and User B. User A is in an enterprise, which has a address and port restricted NAT. User B is on the public internet. There is a TURN+ STUN server deployed INSIDE the enterprise NAT. The NAT has a static set of ports forwarded to the internal TURN server (say, 100 ports). The TURN server is configured with those ports. So, when user A talks to the TURN server it gets an address and port on the *public* side of the NAT, with a preconfigured port forwarding rule. Indeed, the client is configured with two TURN servers. Both are physically the same TURN server. However, when talking to one instance the client gets the public address. When talking to the other instance it gets a private address inside the NAT. The ice process ends up selecting the public address given out by the TURN server usage.]

5. Intercepting Intermediary (B2BUA)

[Editors Note: TODO - a detailed example demonstrating how a B2BUA can obtain STUN/TURN addresses for the purpose of allocating to

Clients. This example shows how intermediaries can control the flow of media without having to directly access SDP on the signalling plane.

6. IPv4-IPv6 Transition

This section describes how IPv6-only SIP user agents can communicate with IPv4-only SIP user agents.

6.1. IPv4-IPv6 Transition for SIP Signalling

IPv4-IPv6 translations at the SIP level usually take place at dual-stack proxies that have both IPv4 and IPv6 DNS entries. Since this translations do not involve NATs that are placed in the middle of two SIP entities, they fall outside the scope of this document. A detailed description of this type of translation can be found in [19]

6.2. IPv4-IPv6 Transition for Media

Figure 30 shows a network of IPv6 SIP user agents that has a relay with a pool of public IPv4 addresses. The IPv6 SIP user agents of this IPv6 network need to communicate with users on the IPv4 Internet. To do so, the IPv6 SIP user agents use TURN to obtain a public IPv4 address from the relay. The mechanism that an IPv6 SIP user agent follows to obtain a public IPv4 address from a relay using TURN is the same as the one followed by a user agent with a private IPv4 address to obtain a public IPv4 address. The example in Figure 31 explains how to use TURN to obtain an IPv4 address and how to use the ANAT semantics [17] of the SDP grouping framework [8] to provide both IPv4 and IPv6 addresses for a particular media stream.

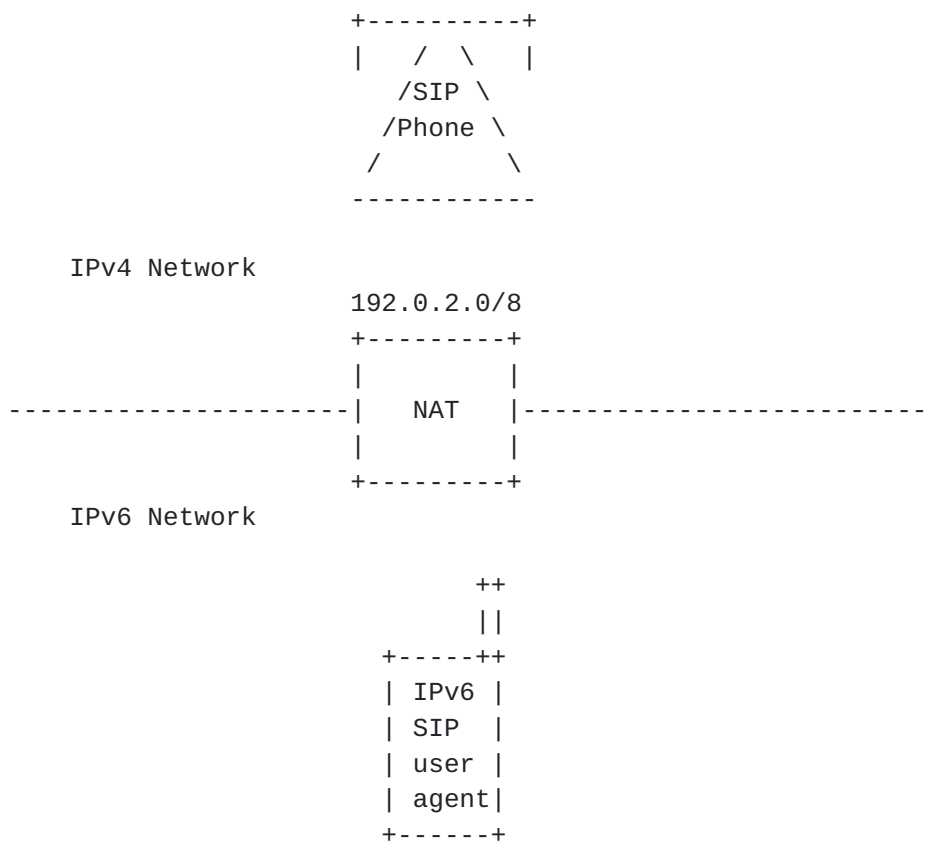


Figure 30: IPv6-IPv4 transition scenario

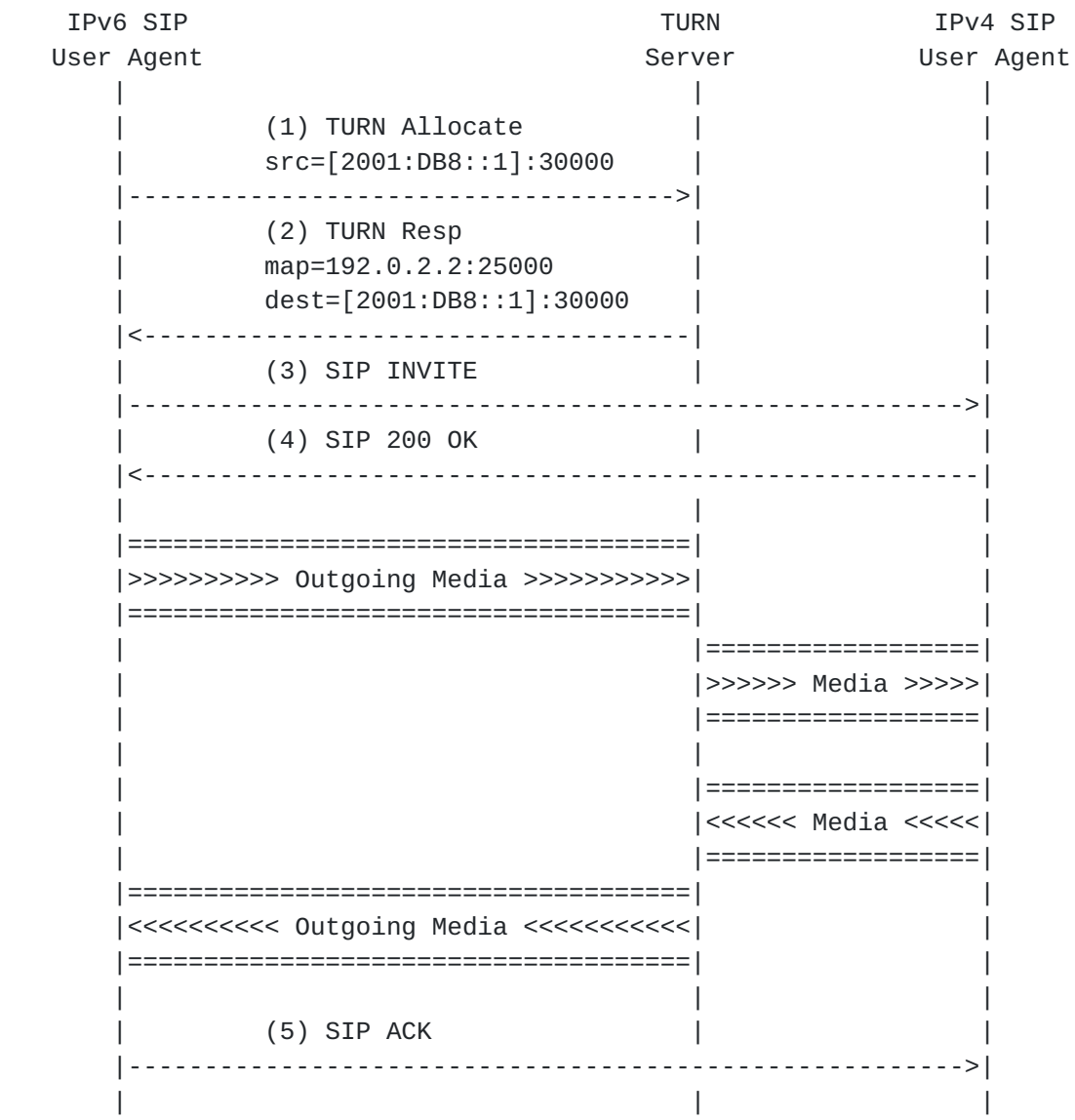


Figure 31: IPv6-IPv4 translation with TURN

- The IPv6 SIP user agent obtains a TURN-derived IPv4 address by issuing a TURN allocate request (1). The TURN server generates a response that contains the public IPv4 address. This IPv4 address maps to the IPv6 source address of the TURN allocate request, which the IPv6 address of the SIP user agent. This results in any traffic being sent to the IPv4 address provided by TURN server (192.0.2.2:25000) will be redirected to the IPv6 address of the SIP user agent ([2001:DB8::1]:30000).
- The TURN-derived address (192.0.2.2:25000) arrives back at the originating user agent (2). This address can then be used in the SDP for the outgoing SIP INVITE request. The user agent builds two media lines, one with its IPv6 address and the other with the

IPv4 address that was just obtained. The user agent groups both media lines using the ANAT semantics as shown below (note that the RTCP attribute in the IPv4 media line would have been obtained by another TURN-derived address which is not shown in the call flow for simplicity).

```
v=0
o=test 2890844342 2890842164 IN IP6 2001:DB8::1
t=0 0
a=group:ANAT 1 2
m=audio 20000 RTP/AVP 0
c=IN IP6 2001:DB8::1
a=mid:1
m=audio 25000 RTP/AVP 0
c=IN IP4 192.0.2.2
a=rtcp:25001
a=mid:2
```

- On receiving the INVITE request, the user agent server rejects the IPv6 media line by setting its port to zero in the answer and starts sending media to the IPv4 address in the offer. The IPv6 user agent sends media through the relay as well, as shown in Figure 31.

7. ICE with RTP/TCP

[Editors Note: TODO - a detailed example will be included on using ICE with RTP/TCP - as define in [18]

8. Acknowledgments

The authors would like to thank the members of the IETF SIPPING WG for their comments and suggestions. Detailed comments were provided by Francois Audet, kaiduan xie and Hans Persson.

9. References

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

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

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

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

[8] Camarillo, G., Eriksson, G., Holler, J., and H. Schulzrinne, "Grouping of Media Lines in the Session Description Protocol (SDP)", RFC 3388, December 2002.

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

[10] Rosenberg, J., "Simple Traversal of UDP Through Network Address Translators (NAT) (STUN)", draft-ietf-behave-rfc3489bis-03 (work in progress), March 2006.

[11] Audet, F. and C. Jennings, "NAT Behavioral Requirements for Unicast UDP", draft-ietf-behave-nat-udp-07 (work in progress), June 2006.

[12] Rosenberg, J., "Obtaining Relay Addresses from Simple Traversal of UDP Through NAT (STUN)", draft-ietf-behave-turn-00 (work in progress), March 2006.

[13] Jennings, C. and A. Hawrylyshen, "SIP Conventions for UAs with Outbound Only Connections", draft-jennings-sipping-outbound-01 (work in progress), February 2005.

[14] Rosenberg, J., "Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)", draft-ietf-sip-gruu-09 (work in progress), June 2006.

[15] Wing, D., "Symmetric RTP and RTCP Considered Helpful", draft-wing-mmusic-symmetric-rtprtcp-01 (work in progress), October 2004.

[16] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", draft-ietf-mmusic-ice-08 (work in progress), March 2006.

[17] Camarillo, G., "The Alternative Network Address Types Semantics (ANAT) for theSession Description Protocol (SDP) Grouping Framework", draft-ietf-mmusic-anat-02 (work in progress), October 2004.

[18] Rosenberg, J., "TCP Candidates with Interactive Connectivity Establishment (ICE)", draft-ietf-mmusic-ice-tcp-00 (work in progress), March 2006.

9.2. Informative References

[19] Camarillo, G., "IPv6 Transcition in the Session Initiation Protocol (SIP)", draft-camarillo-sipping-v6-transition-00 (work in progress), February 2005.

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@cisco.com

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: Gonzalo.Camarillo@ericsson.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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