Internet Engineering Task Force SIPPING WG
Internet Draft Rosenberg,Mahy,Sen
draft-rosenberg-sipping-nat-scenarios-00.txt dynamicsoft,Cisco,Nortel
November 14, 2001
Expires: March 2002

NAT and Firewall Scenarios and Solutions for SIP

STATUS OF THIS MEMO

Abstract

The problem of firewall and nat traversal for SIP is a complex one, due, in part, to the large number of different scenarios and the multitude of solutions developed to solve them. In this draft, we enumerate the various scenarios which can arise, and for each, point to some of the existing solutions for that scenario, and present call flows and explanations for how it works.

1 Introduction

The problem of firewall and NAT traversal is one that receives a very large amount of attention. It is particularly troublesome for interactive communications, such as those established and managed by SIP [1]. A host of solutions have been proposed and discussed, including the Simple Traversal of UDP Through NAT (STUN) protocol

[2], Traversal Using Relay NAT (TURN) [3], SIP ALGs [4] [5] [6], the MIDCOM protocol [7], SDP extensions for NAT [8], SIP extensions for NAT [9], RSIP [10] [11], MGCP controlled firewalls [12], tunnels of various flavors [13], and even an April Fools RFC, RFC 3093 [14], which many people have taken quite seriously.

This is further complicated by the variety of scenarios that are frequently discussed, including service providers, enterprise, centrex, residential, nats alone, nats and firewalls, firewalls alone, and so on.

All of this is the source of unending confusion. For the novice wishing to understand how to solve the "nat problem", it is a nearly insurmountable hurdle to sort through these drafts and scenarios, to put together a coherent view of the options at hand.

This draft is a first attempt to resolve that problem. We enumerate what we believe is a comprehensive set of scenarios that have been discussed, and for each scenario, describe some of the solutions and the drafts that would be needed to make them work. Our solution sets are not complete, as they omit discussion of approaches such as RSIP and VPN. Those are coming in a future revision. We have also omitted a good comparison of the solutions, which will also be present in a future revision.

The scenarios that are considered include (1) a user behind a residential NAT/FW using a public service provider's VoIP service (Section 2), (2) a user within an enterprise that wants to use the services of a VoIP provider on the public Internet, but whos enterprise is not supporting the service (Section 3), (3) a user within an enterprise that is deploying VoIP itself, and is not using any VoIP provider (Section 4), and (4) a user within an enterprise that is providing VoIP services, but is doing so by outsourcing the service to a public provider. This is effectively a Centrex approach. (Section 5).

2 Scenario I: Residence with single NAT

In this scenario, a user has a broadband connection to the Internet, using a cable modem or DSL, for example. In order to provide security, or to run multiple machines, the user has purchased an off-the-shelf "DSL Router" as they are called. These devices, manufactured by companies such as Linksys, Netgear, 2wire, and

Netopia, generally include a NAT, simple firewall, DHCP server and client, and a built in ethernet switch of some sort. The firewall generally allows all outgoing traffic, but disallows incoming traffic unless specific port forwarding or a DMZ host has been configured. The NAT treatment of UDP in these boxes varies. The most common types

appear to be full-cone and restricted cone. See [2] for a definition of these terms.

The user in this scenario wishes to use a communications service from a retail provider, such as net2phone or deltathree, for example. The connection between the user and the provider is through the cable modem or DSL, through the public Internet. The user may have multiple PCs in their home accessing this service, but they are not related in any way. This scenario also includes the case where its not a PC, but a standalone SIP phone. In this case, the provider might be providing some kind of second line VoIP service.

This scenario is depicted in Figure 1.

2.1 Solution I: Configuration

The most simplistic solution for this case is to configure the NAT and the PC or softphone to allow for service.

Most residential NATs allow the NAT to be configured with a DMZ host. This is a host that will receive all incoming packets that are not associated with some kind of outgoing connection established previously. The NAT also allows the user to find out the IP address that was assigned to them from their cable modem or DSL provider.

The procedure is then simple:

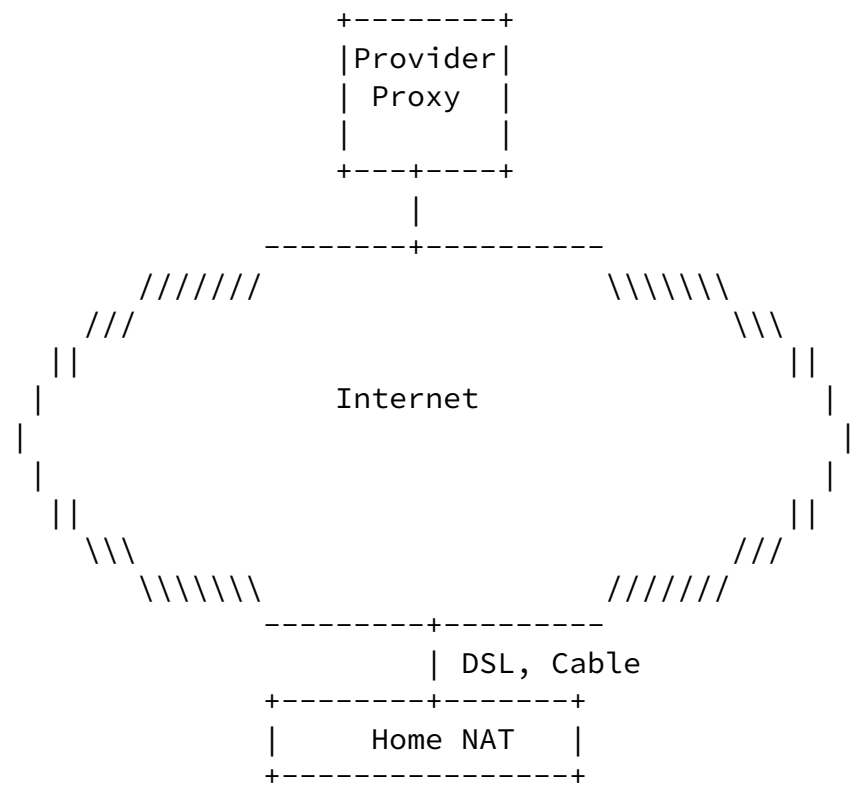
1. Determine the IP address assigned to the user by the cable or DSL provider, by looking through the residential NAT configuration. Call this the "public address".

2. Determine the IP address assigned to the phone that the user wishes to use. This address is on the home LAN, assigned by the DHCP server running in the residential NAT. Call this the "phone address".

3. Enable the DMZ host configuration on the residential NAT. Set the DMZ host to be equal to the phone address. This means that the phone will receive all incoming traffic.

4. Configure the PC softphone or hardphone to use the public address in all SIP and SDP messages it builds. Many VoIP clients can be configured in this way.

Thats it! This configuration effectively makes the NAT transparent; all packets in go to the phone without being natted, and the phone has the public address that will get routed to it.



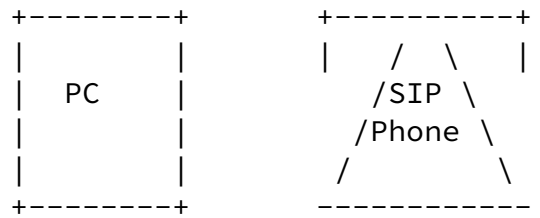


Figure 1: Residence with Single NAT

A PC to phone call flow for this case is shown in Figure 2. Since the PC client is configured with the IP address on the public side of the NAT (1.2.3.4), that is the address inserted into the SDP of the INVITE (message 1). When the proxy sends the 200 OK response to the PC (message 4), it sends it to the source IP address where the request came from (1.2.3.4), but the port in the Via header (5060).

This is received by the NAT. Since there is no mapping for this, the IP address is rewritten to the DMZ host's address (10.0.1.1), and sent there. This is exactly what we want, since the PC is listening on 5060 for the responses. The media from the gateway to the PC works in a similar way (line 9). Its sent from the gateway to the IP address and port in the SDP (1.2.3.4:8876). The NAT receives this. Since there is no binding, the IP address is rewritten to the DMZ host (1.2.3.4) and the RTP packets forwarded there.

Receiving calls will work in a similar way. The Contact header in the registrations from the PC client or hard phone will include the public address (1.2.3.4), and therefore incoming calls are routed there.

The benefits of this solution are that it works without additional software changes. However, the drawbacks are many. It requires complex configuration which is certainly beyond the capabilities of most users. Secondly, it only works with a single phone at a time talking. If a different phone is used, the configuration would need to change. Third, it will open up the phone to a variety of DoS attacks, since all other kinds of traffic will be let in towards the

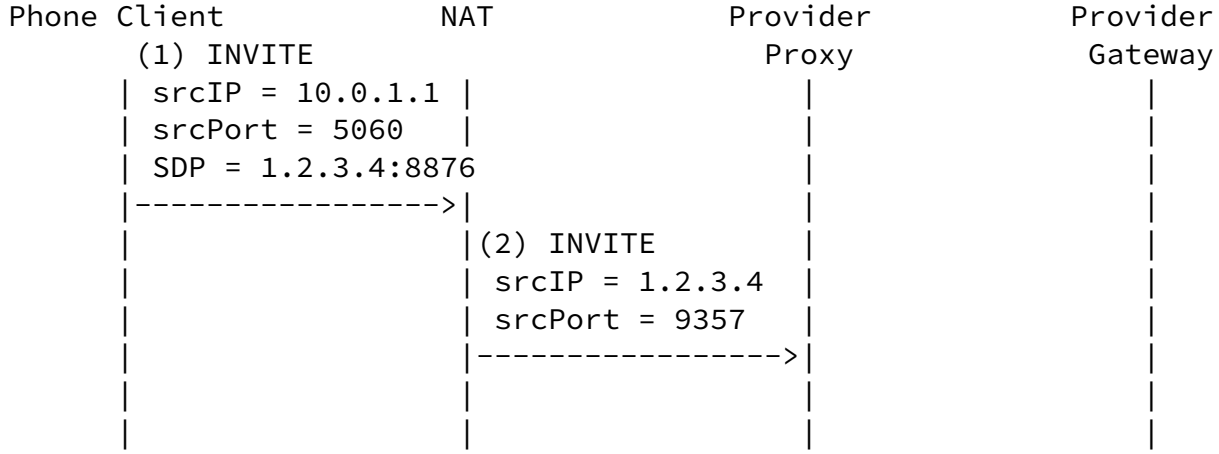
phone.

2.2 Solution II: Stun in Client

The second solution for this scenario is to upgrade the client phone or PC application to include support for the STUN protocol [2], and optionally the SDP extensions for NAT [8], SDP extensions for connection oriented media [15], and SIP extensions for NAT [9]. The latter three are all optional, as the solution can work without them. We discuss the pros and cons of having or not having these. This solution also requires the service provider to deploy the stun server described in [2].

The stun protocol allows a client to discover whether it is behind a nat, and what type of nat it is behind. In the case of three of the four nat types described in [2] (all but the symmetric NAT), stun will provide the client with its public IP address.

When the client first starts up, the first thing it will do is to use STUN to determine the type of NAT it is behind. This check need only be done on startup; the type of NAT will not generally change unless the user upgrades or replaces their NAT, in which case a boot of the phone or client would be needed. The procedure on startup is the discovery procedure that is described in Section 9.1 of the stun protocol. This involves messaging exchanges between the stun client and its stun server. The reader is referred to that document for details.



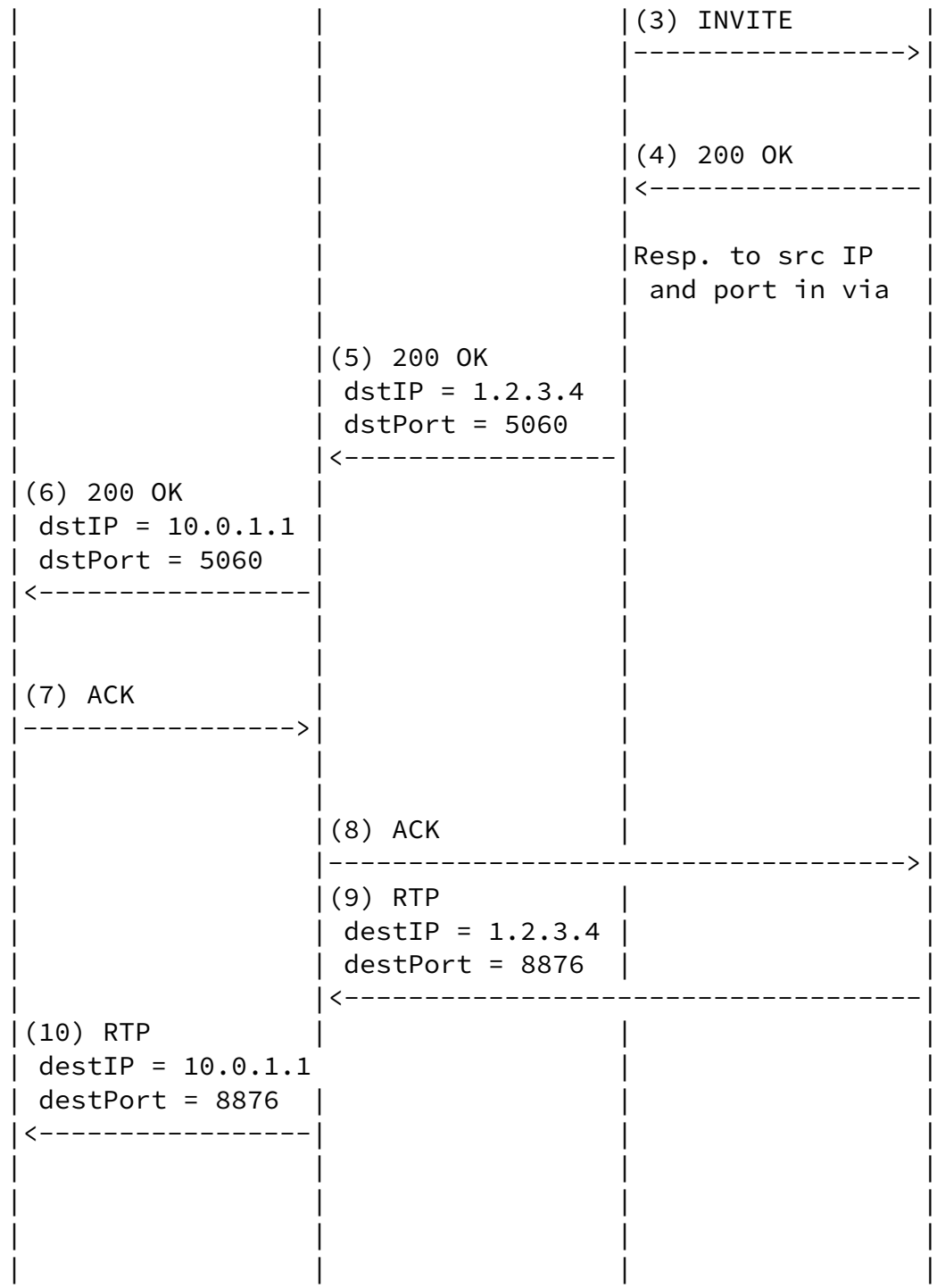


Figure 2: Configuration solution

Once the type of NAT is discovered, operation from there on depends

on the result of the discovery. If the user discovers that they are not behind any NAT or firewall at all, then no special processing is required. If, however, they discover that they are behind a NAT, the processing depends on whether the NAT is symmetric or not.

2.2.1 Not Symmetric NAT

In this scenario, the client has discovered that it is behind a full cone or restricted cone NAT. This includes the case where there are multiple NATs (its common for cable modem providers to NAT their entire networks, so this NAT might exist in addition to the NAT in the user's home), but the most restrictive type is full cone or restricted cone.

2.2.1.1 Registration

The first step is for the client to register. The recommended procedure for that is to use the SIP extensions for NAT [9], along with STUN. The flow is shown in Figure 3.

First, the client uses a STUN query to its provider's STUN server. This will provide it with an IP address and port on which it can receive messages. The client will use the address and port in the STUN response (message 4) to receive incoming SIP messages. In order to do that, it includes this address and port as the Contact header of a REGISTER message that it sends to its provider's proxy (message 5), preferably over TCP. This REGISTER also optionally includes the Translate header defined in [9]. This header asks the server to ignore the Contact header, and instead send all requests on the TCP connection the register is being sent over. The Translate header contains the value of the Contact header which is to be ignored.

If the proxy supports SIP extensions for NAT [9], the response to the REGISTER (message 7) will contain a Contact header different from the one provided in the request. In this case, the client knows that the address allocated through STUN will not be needed, and there is no need to refresh that address. If, on the other hand, the REGISTER response contains the same Contact provided in the request, the client knows that the Translate header was not supported. In this case, the client can close the TCP connection used to send the REGISTER. It will need to refresh the binding learned from STUN by re-sending the STUN query every minute or so. It will also need to listen for incoming SIP messages on this address and port.

If the client elects not to implement the Translate header, it simply omits it from the request in the flow above. In that case, the client

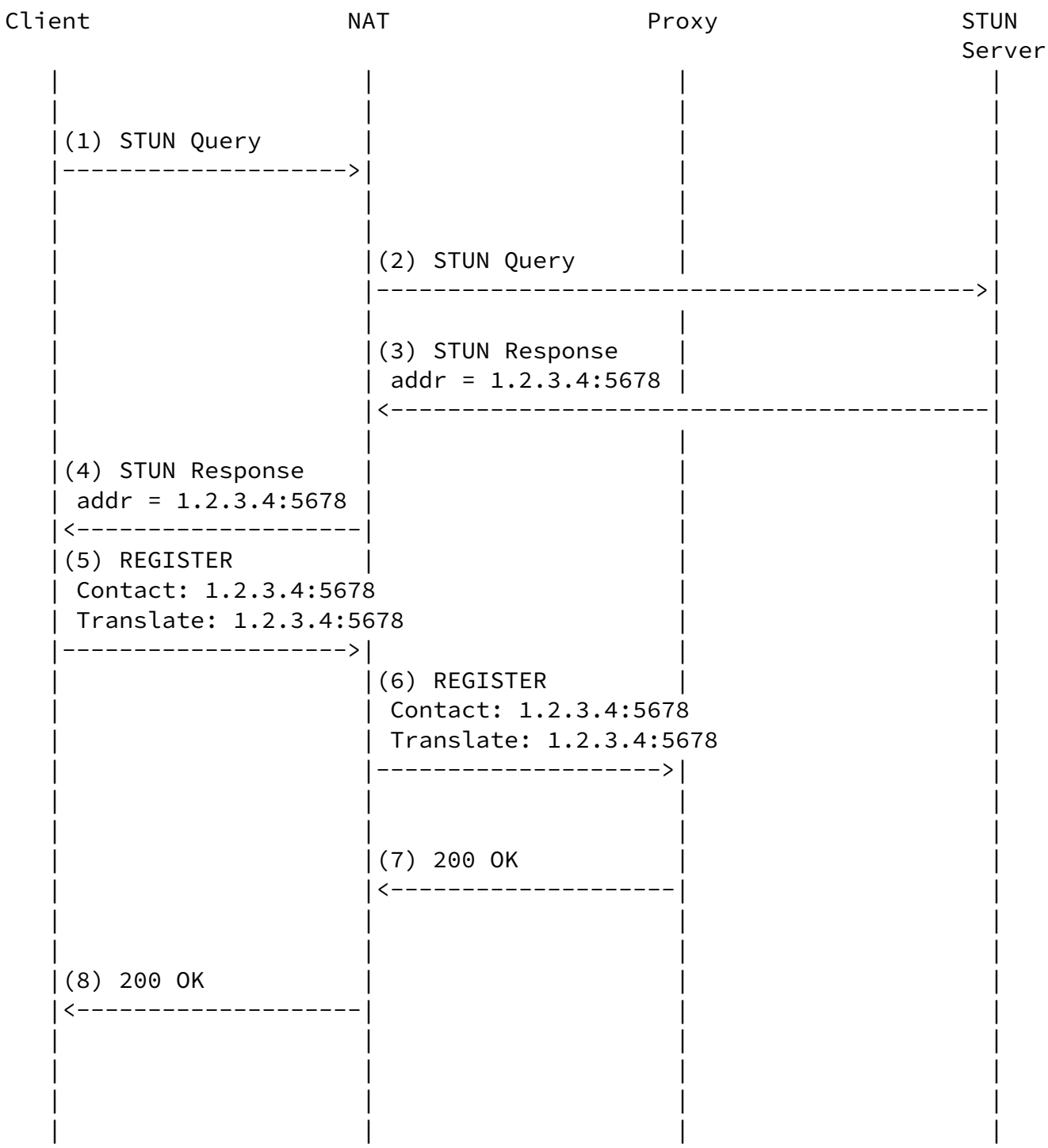


Figure 3: Register flow, not symmetric NAT

will have to use the address obtained from STUN.

The usage of the Translate header, along with a TCP connection to the

server, is preferred in order to improve overall scale. This solution requires the fewest elements to be involved, and requires the least amount of message traffic. Using STUN for incoming SIP requests will require continuously refreshing the binding learned through STUN, which is very intensive (a refresh at least every minute).

However, if TCP is not available, the alternative is to use UDP registrations. In that case, the flow is identical, but the server has to support the rport parameter specified in [9], otherwise the SIP responses will not properly go through the NAT.

2.2.1.2 Initiating a Session

The next operation to consider is how the client makes a call. The flow for this process is shown in Figure 4. We assume the simple case of a single audio stream. In that case, the client will require two IP addresses and ports - one to receive RTP, and the other, to receive RTP. As a result, it launches two STUN queries (messages 1-4 and message 5-8), resulting in two address/port pairs. Unfortunately, the RTP and RTCP ports are no longer adjacent. This will require the SDP in the INVITE constructed by the client to include the address for RTCP as a separate element. An SDP extension for doing this is defined in [8]. If this extension is not supported by the client (or the called party), the result is that RTCP may not be transmitted for the session. This is bad, but not catastrophic, as the call can proceed without it.

The client will also need to populate the IP address in the Contact header of the INVITE. This will generally be the same Contact address which was registered from the process in Section 2.2.1. This should always be the value from the Contact header in the response to the REGISTER request.

The final component of the INVITE is the support for connection oriented media [15]. Connection oriented media are media that run over transports that have the notion of a connection that needs to be established. Normally, this means TCP. In the case of TCP, if two

parties want to communicate, one needs to initiate the connection, and the other needs to receive it. Once established, both sides can send over the connection. To support that, the comedia draft [15] defines a new SDP attribute, the a=direction attribute. This attribute can take on the values of passive, active, or both. Passive means that the participant can only be on the receiving end of the connection. Active means it must be the initiating side. Both means that it can do either. Connection oriented operation can also be applied to RTP over UDP. We call this mode of operation "symmetric RTP". In symmetric RTP, the active side sends an RTP packet to the

passive side. When the passive side receives this, it sends RTP packets back to the source address of the RTP packet just received. Thats it.

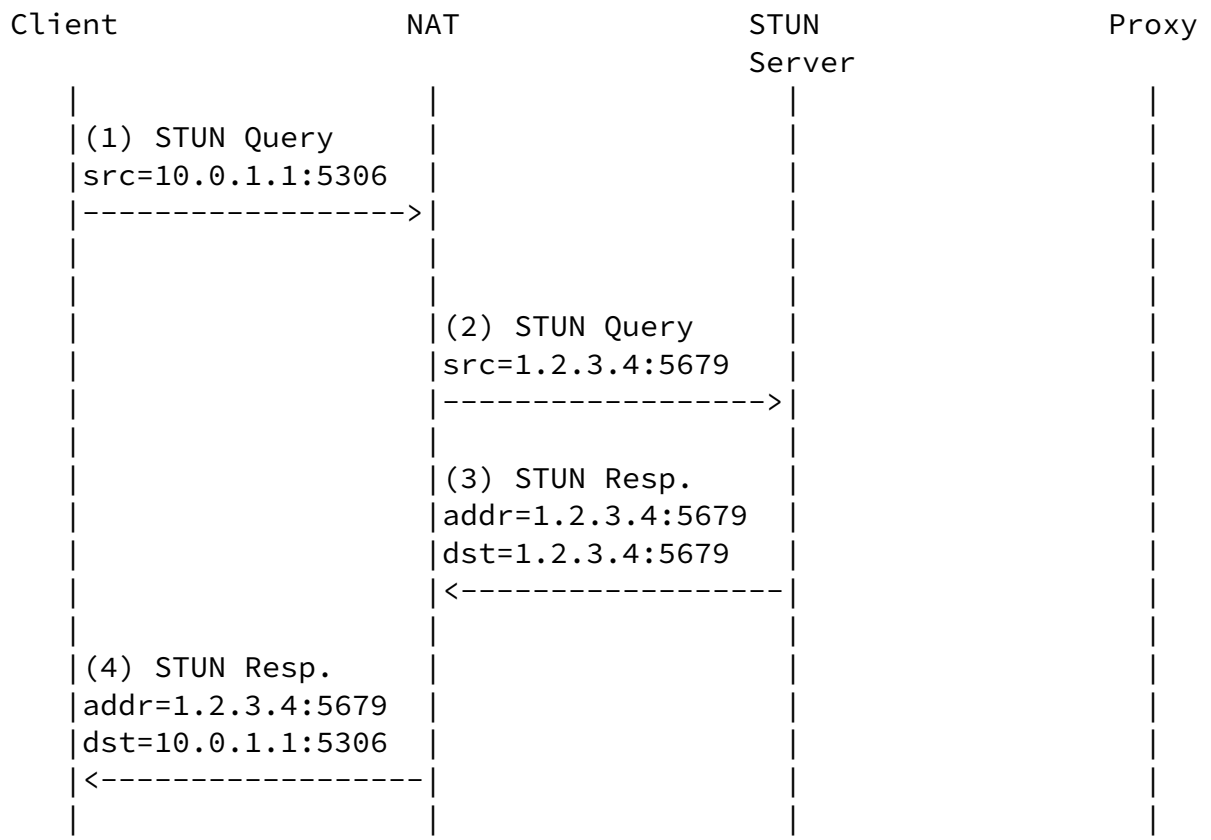
Symmetric RTP has the important property that one side (the active side) does not need to provide an IP address and port to the peer in its SDP, since no packet is sent there. Rather, its sent to the source address of the received RTP packet from the active side. This means that the active side can be behind a symmetric NAT, and still send media directly to its peer without use of some kind of network intermediary. This is good. However, for it to work, the peer needs to support symmetric RTP. So, even though in the case currently under consideration, the client is not behind a symmetric NAT, it should ideally support symmetric RTP just in case its peer is behind a symmetric NAT.

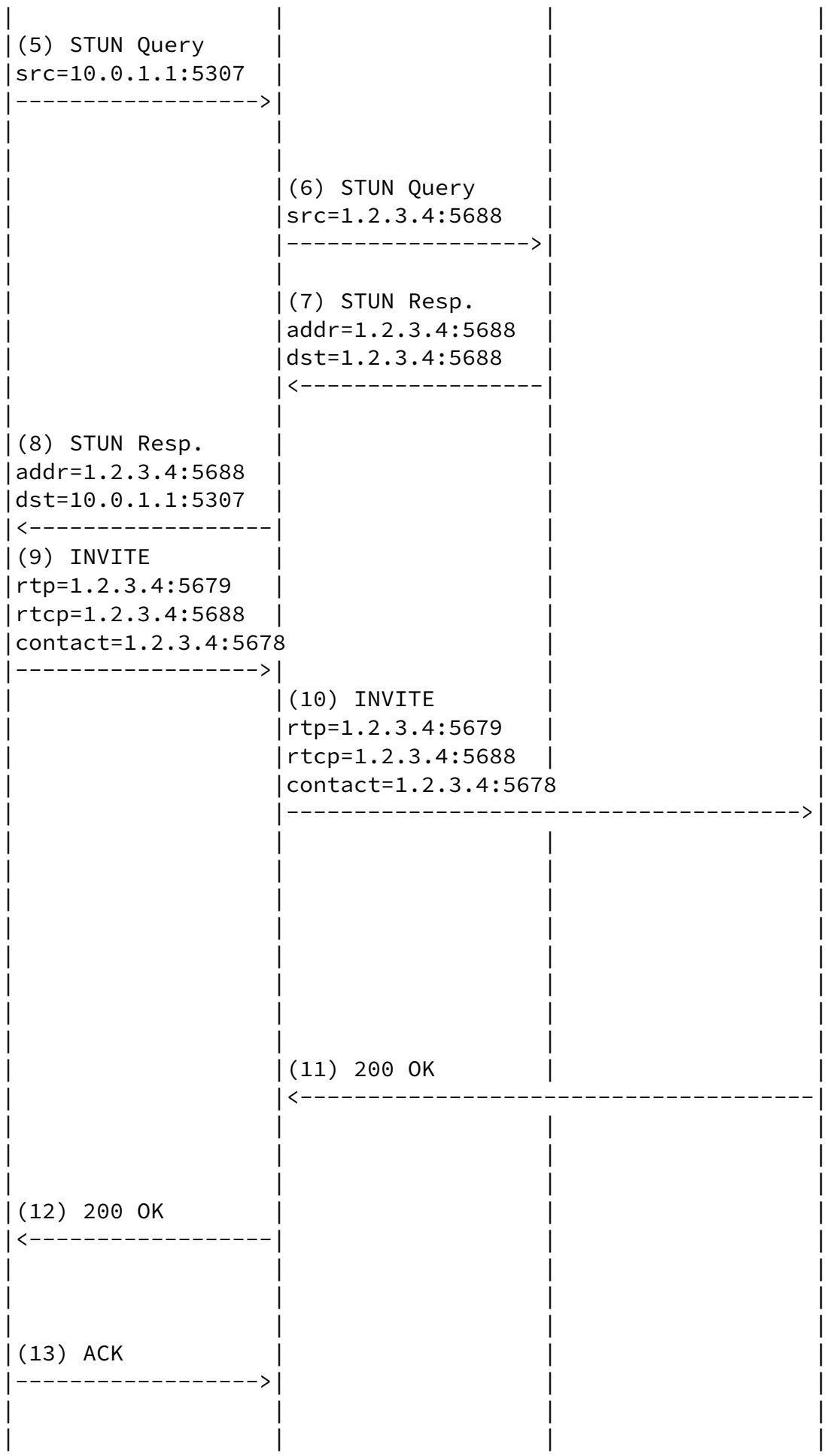
Therefore, if the client supports the comedia extension applied to RTP, it should include the a=direction:both attribute in its SDP. If this extension is not supported, things will still work, but the call may be routed through an intermediary if the peer is behind a NAT.

The Via header contains an IP address as well. This address will never be used, in fact, except for failure conditions at the proxy. Therefore, the address should contain the same address used in the contact header. The SDP o line also has an IP address, used for identification purposes only. Its value is arbitrary, but should be the contact address also, to prevent leakage of any private addresses outside the network.

The outgoing INVITE (message 9) might look like:

```
INVITE sip:user@domain SIP/2.0
From: sip:user@school.edu;tag=88asd
To: sip:user@domain
Call-ID: 98asd6asd60099
CSeq: 987769 INVITE
Via: SIP/2.0/TCP 1.2.3.4
Contact: sip:1.2.3.4:5678
Content-Type: application/sdp
Content-Length: ...

v=0
o=aa 2890844526 2890842807 IN IP4 1.2.3.4
c=IN IP4 1.2.3.4
t=0 0
m=audio 5679 RTP/AVP 0
a=rtcp:5688
```


(5) STUN Query
src=10.0.1.1:5307
(6) STUN Query
src=1.2.3.4:5688
(7) STUN Resp.
addr=1.2.3.4:5688
dst=1.2.3.4:5688
(8) STUN Resp.
addr=1.2.3.4:5688
dst=10.0.1.1:5307
(9) INVITE
rtp=1.2.3.4:5679
rtcp=1.2.3.4:5688
contact=1.2.3.4:5678
(10) INVITE
rtp=1.2.3.4:5679
rtcp=1.2.3.4:5688
contact=1.2.3.4:5678
Proxy forwards
INVITE,
response
comes
(11) 200 OK
(12) 200 OK
(13) ACK

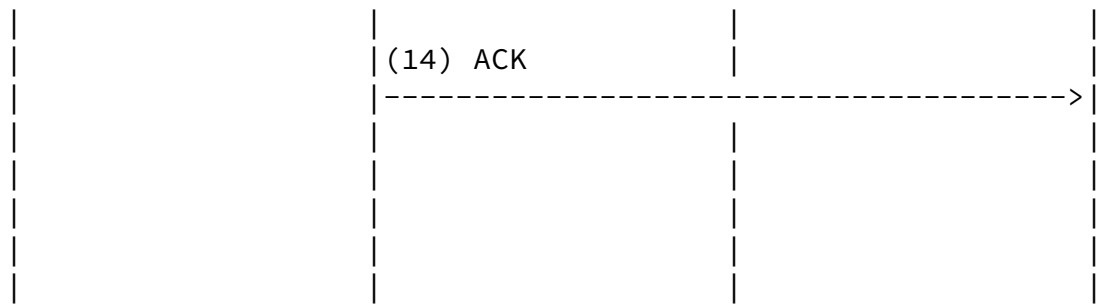


Figure 4: Initiating a session, not symmetric NAT

a=direction:both

The INVITE goes through the NAT, and is received at the proxy (message 10). The proxy forwards it on, and receives a 2xx response. This response is passed through the nat (message 11) and arrives at the client (message 12).

The proxy should have record-routed, in order to ensure that the messaging flows through it (as its holding the only signaling connection to the client). As a result, the ACK should be sent over that TCP connection to the proxy (the same for UDP; it would be sent from the same socket, to the same place the REGISTER was sent).

If symmetric RTP is being used, the SDP in the response will contain an a=direction attribute, in which case the procedures of [15] would apply. If there is no symmetric RTP in use, the client sends media to the address(es) provided in the 200 OK. The media should be sent from the same socket that is being used to receive (the two learned through STUN), although this is not strictly needed.

2.2.1.3 Receiving an Invitation to a Session

The call flow for receiving an invitation to a session is shown in Figure 5. The flow resembles the one for the outgoing INVITE in many ways. When the INVITE arrives at the client, it will need to perform two STUN queries for each media stream (messages 3-6 for the RTP address and messages 7-10 for the RTCP address in the case of a single media stream). This is to obtain the addresses to use for

receiving media.

The client also checks for the comedia a=direction attribute (assuming this extension is supported). If it is, the guidelines in [15] are followed to set the direction attribute in the 200 OK. If it was a=active in the INVITE, this means that the peer is likely behind a symmetric NAT. The direction attribute in the response will then be a=passive. If the client doesn't support comedia, the attribute is ignored and no direction attribute is placed in the response.

As with the initiation case, the SDP extensions for RTCP [8] will need to be used to explicitly specify the RTCP address. The same guidlines for selecting the Contact, Via, and o line as described in the initiation case apply here as well. The result is a 200 OK response (assuming the call is accepted, of course), which looks like (message 11):


```
SIP/2.0 200 OK
From: sip:user@school.edu;tag=88asd
To: sip:user@domain;tag=99as8a
Call-ID: 98asd6asd60099
CSeq: 987769 INVITE
Via: SIP/2.0/TCP 1.9.2.2
Via: SIP/2.0/UDP 82.3.4.5
Via: SIP/2.0/TCP 1.2.3.4
Contact: sip:9.8.7.6:7766
Content-Type: application/sdp
Content-Length: ...

v=0
o=aa 2890844526 2890842807 IN IP4 9.8.7.6
c=IN IP4 9.8.7.6
t=0 0
m=audio 9988 RTP/AVP 0
a=rtcp:8877
a=direction:passive
```

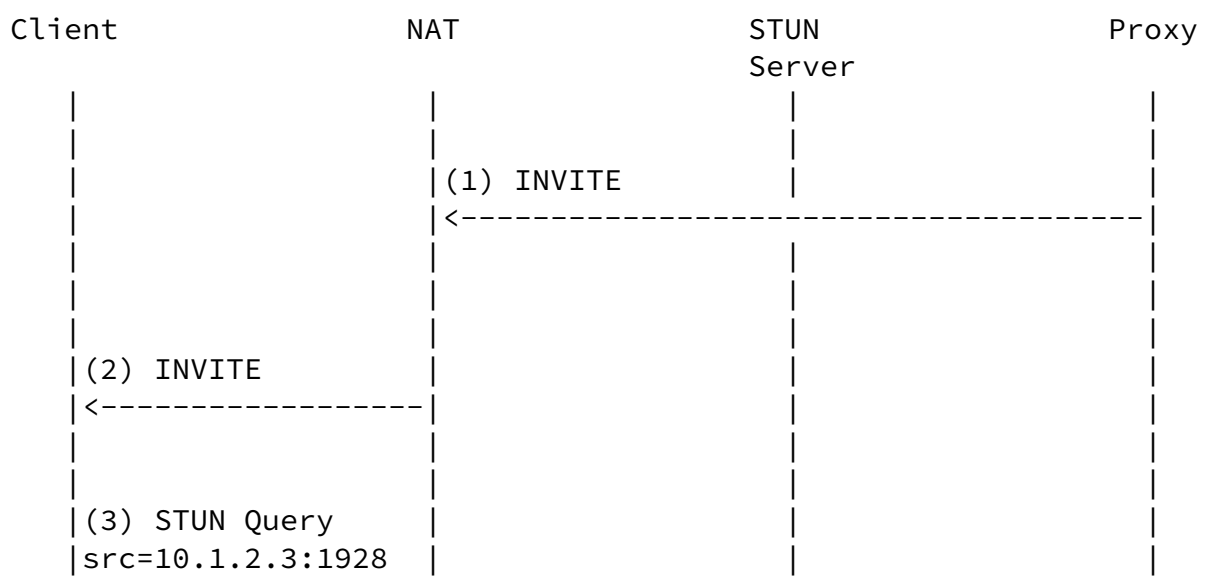
The client will receive the ACK over the same connection it received the INVITE, since its proxy will record route.

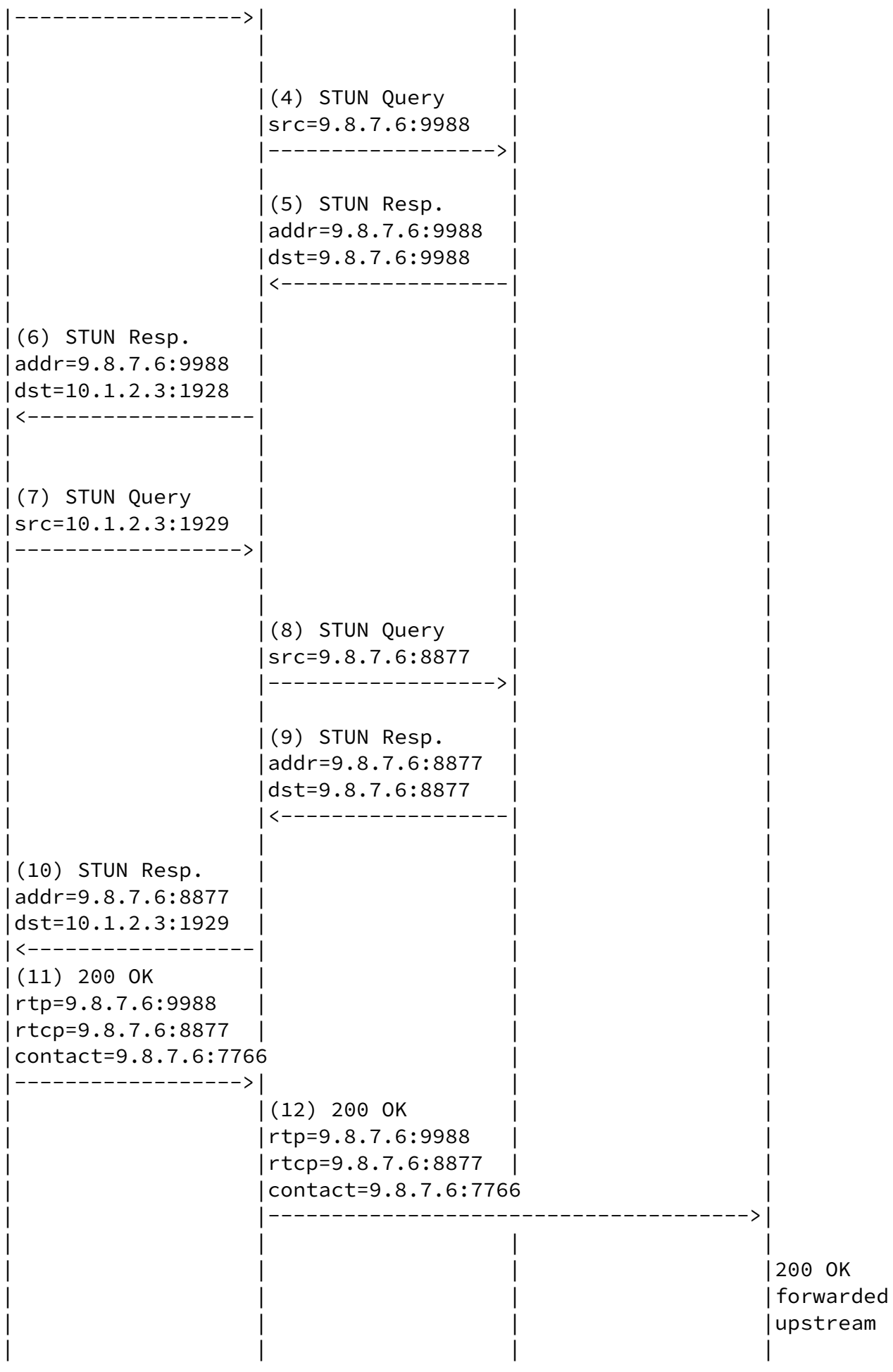
2.2.1.4 Media Flow

Finally, let us consider the actual flow of media in the examples above. The examples had a call placed from user@school, behind a NAT with public IP 1.2.3.4, calling user@domain, behind a NAT with public IP 9.8.7.6. user@school used the comedia extensions, with a direction attribute of both, and user@domain answered with a direction of passive. The result is that user@school will act as active, and send the first RTP (and RTCP) packet. This will flow through its NAT, the NAT of user@domain, and then to user@domain.

The flow of the media for this case is shown in Figure 6. The RTCP is not shown, but would work in an identical fashion.

This flow will work perfectly for full cone, but will not work as written for restricted cone. Thats because NAT 2, assuming its restricted cone, will not allow the RTP packet from user@school (message 2) in. Thats because user@domain has not yet sent a packet to 1.2.3.4 yet. Therefore, user@domain needs to "prime" its NAT by sending a packet to 1.2.3.4:5679 (it will know this address from the INVITE it received). Thus, it needs to send a packet even though it has indicated it wants to be the passive side of the symmetric RTP




(4) STUN Query
src=9.8.7.6:9988
(5) STUN Resp.
addr=9.8.7.6:9988
dst=9.8.7.6:9988
(6) STUN Resp.
addr=9.8.7.6:9988
dst=10.1.2.3:1928
(7) STUN Query
src=10.1.2.3:1929
(8) STUN Query
src=9.8.7.6:8877
(9) STUN Resp.
addr=9.8.7.6:8877
dst=9.8.7.6:8877
(10) STUN Resp.
addr=9.8.7.6:8877
dst=10.1.2.3:1929
(11) 200 OK
rtp=9.8.7.6:9988
rtcp=9.8.7.6:8877
contact=9.8.7.6:7766
(12) 200 OK
rtp=9.8.7.6:9988
rtcp=9.8.7.6:8877
contact=9.8.7.6:7766
200 OK
forwarded
upstream

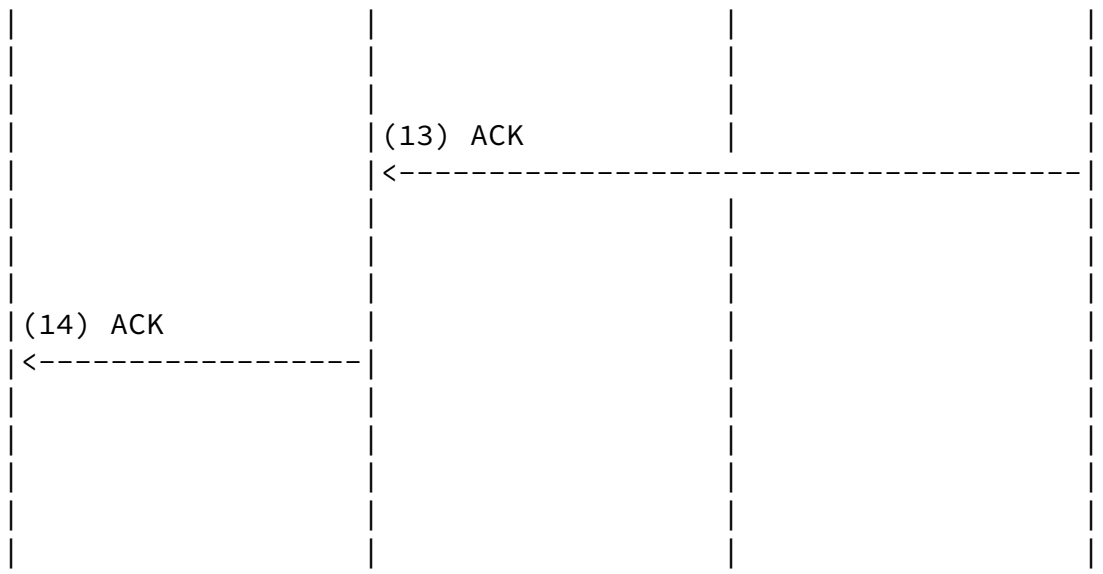
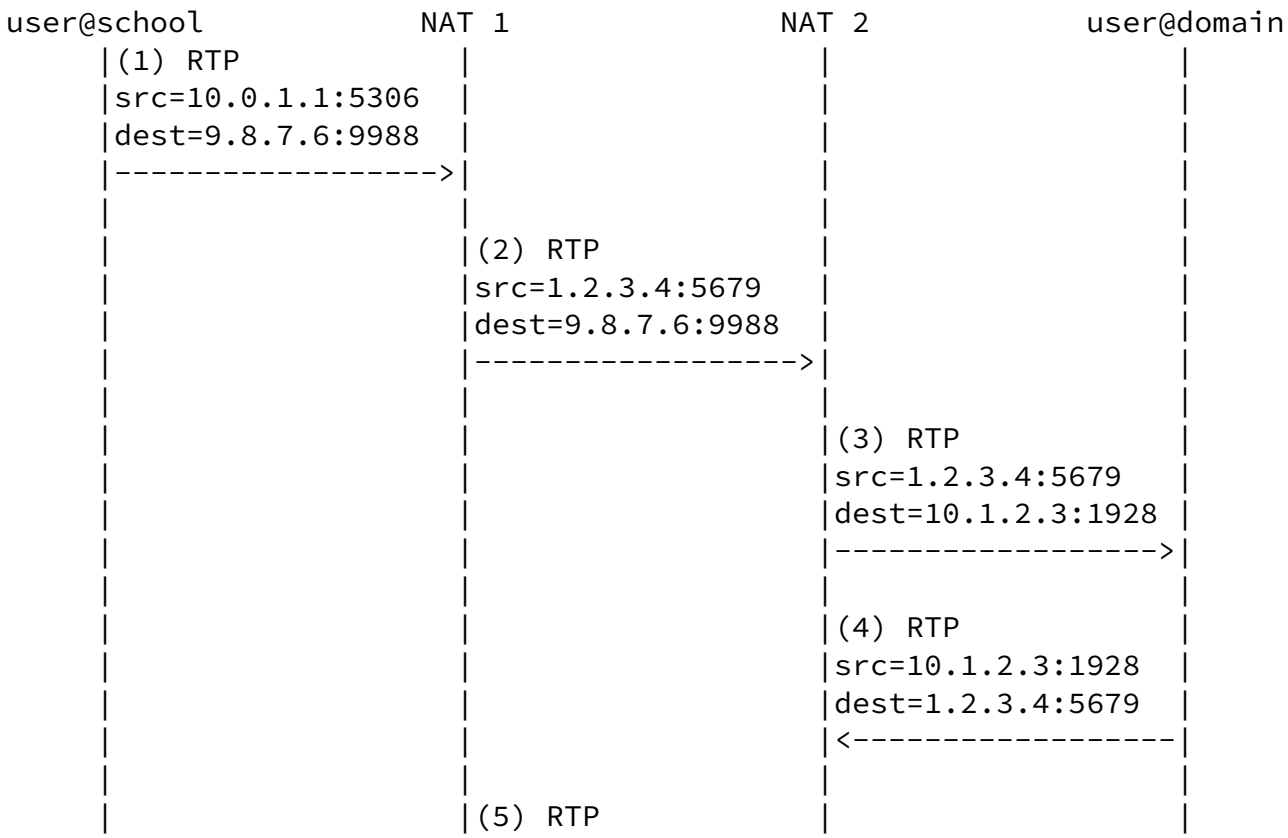


Figure 5: Incoming INVITE, not Symmetric NAT



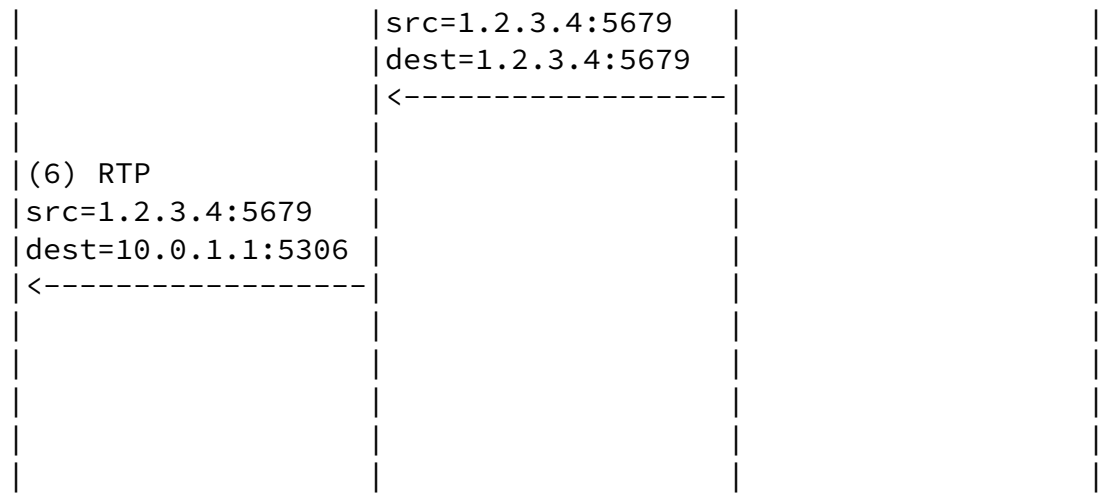


Figure 6: Media flow for residential case, not symmetric NAT

connection. The most natural way to accomplish this is to actually have users behind restricted cone or port restricted cones respond to a=direction:both with a=direction:both. In that scenario, as described in [15], both sides send a packet to each other, and one of the "connections" is then used. Having both sides send packets to each other is exactly what is needed to allow operation in restricted

cone and port restricted cone nats.

The flow for this case (where both sides indicated a=direction:both) is shown in Figure 7. It is identical to the flow above, but with a reordering of messages. Now, both users send their first packet at the same time, causing the NATs to be "primed". In reality, the packets may not be sent at the same time, so the first few packets from the first sender will be lost until the other user primes their NAT.

Its important to realize that with port restricted cone NAT, the priming will only work when the user sends from the same IP address and port they expect to receive from.

2.2.2 Symmetric NAT

Section 2.2.1 handled the case where the user was in a residence behind a full cone or restricted cone NAT. In this section, we consider the case of a symmetric NAT.

The operation is much like the non-symmetric case. However, instead of using STUN to determine the addresses, a TURN [3] server is used to obtain addresses. Like STUN, TURN will provide a publically routable IP address and port that can be used. However, unlike STUN, this address and port routes to the TURN server, which forwards the packets on the same socket the TURN request was received on. This allows it to operate through more restrictive symmetric NAT. The cost of this approach is routing of media through the service provider network, which can substantially increase latency and loss, and also increases the cost to the provider.

2.2.2.1 Registration

The recommended registration mechanism parallels the procedure for STUN as described in Section 2.2.1. The client will allocate an address from the TURN server, and then include that in the Contact header in the REGISTER, along with the Translate header. In the case of TURN, it is recommended that a TCP address be allocated from the TURN server. This will avoid the need to constantly refresh the binding, as would be the case for UDP. The problem is worse with TURN, since once the client receives an incoming INVITE, refreshes of the binding would cause packets to be delivered to the proxy, and therefore they would need to be complete REGISTER messages themselves in order to be properly processed. This will result in a serious burden on the proxy server.

Assuming TCP, the flow for registration is shown in Figure 8. Before message 1 is sent, there is a standard TCP handshake with the TURN server, not shown. Then, the client sends a TURN Allocate request (message 1) to its TURN server. The response (message 4) contains the address allocated to the client, in this case, 15.1.2.3:5678. So, it uses that address in a REGISTER message, also sent using TCP, to its proxy (message 5).

2.2.2.2 Initiating a Session

The process for initiating a session parallels that for the non-symmetric case in Section 2.2.1. The client will use TURN to allocate two UDP addresses - one for the RTP, and another for the RTCP. The INVITE that is constructed will use the SDP extensions for RTCP [8] to contain the RTCP addresses. The construction of the Via, Contact, and origin lines in the SDP mirrors that for the non-symmetric case.

The main difference is in the use of comedia. For the symmetric case, support for the comedia draft [15] is strongly recommended. If the other participant in the call is not behind a NAT, or is behind a full cone for restricted cone NAT, the use of the TURN server as relay can be avoided. This will improve the voice quality and reduce costs to the provider. In order for its usage to be avoided, the client includes an a=direction:active attribute in its SDP. Note that it still includes the addresses it obtained from the TURN server. These will end up being used if the peer doesn't support comedia, or is behind a symmetric or port restricted cone NAT.

> This usage of a=active differs from the current comedia draft, which says that if a=active is present, you MUSTNOT initiate a connection to it. This is not backwards compatible. The approach described here is. We should work to get this incorporated into comedia to add backwards compatibility.

A flow for this case is shown in Figure 9.

The resulting INVITE generated might look like:

```
INVITE sip:user@domain SIP/2.0
From: sip:user@school.edu;tag=88asd
To: sip:user@domain
Call-ID: 98asd6asd60099
CSeq: 987769 INVITE
Via: SIP/2.0/TCP 15.1.2.3
```

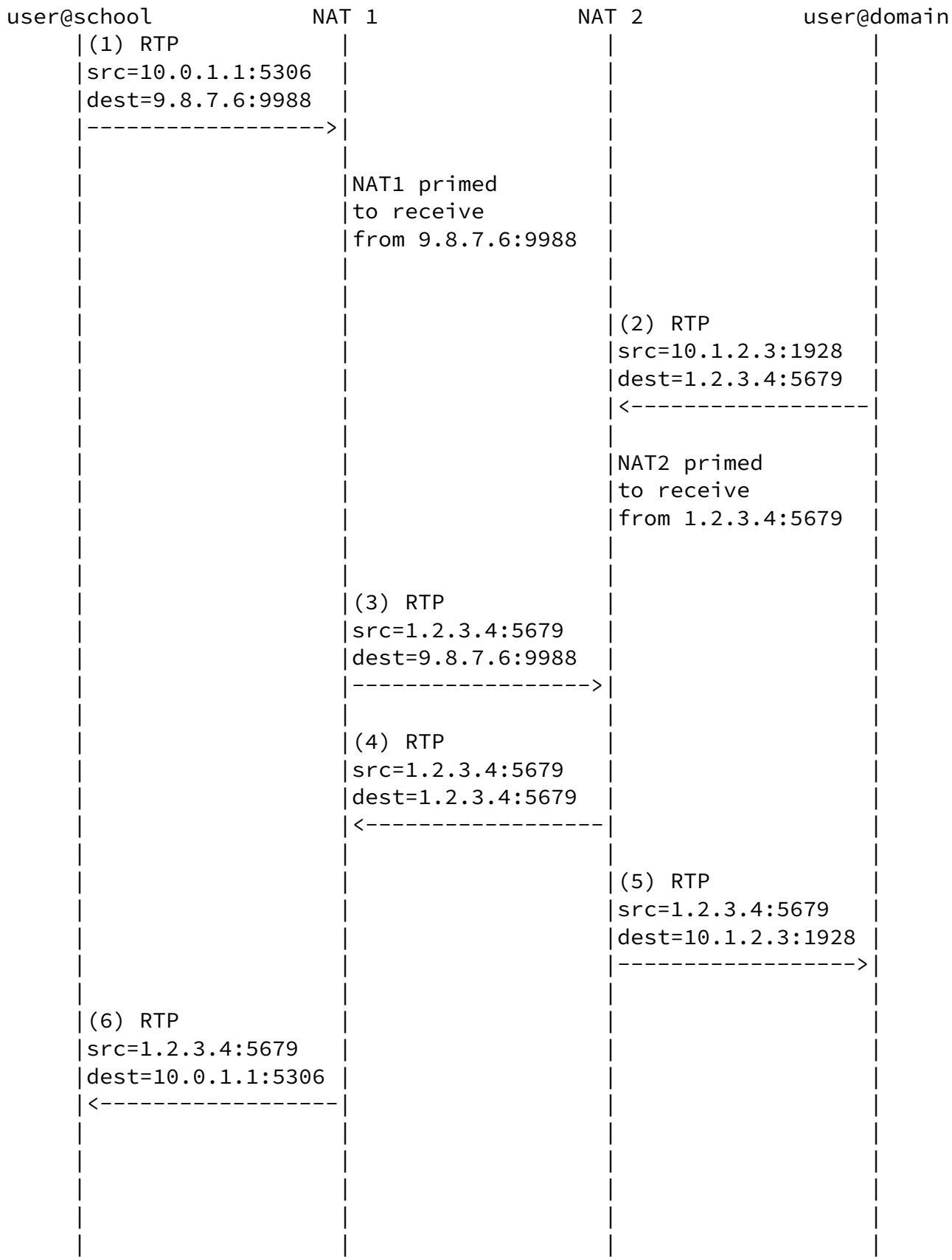


Figure 7: Media flow for residential case, priming the NAT

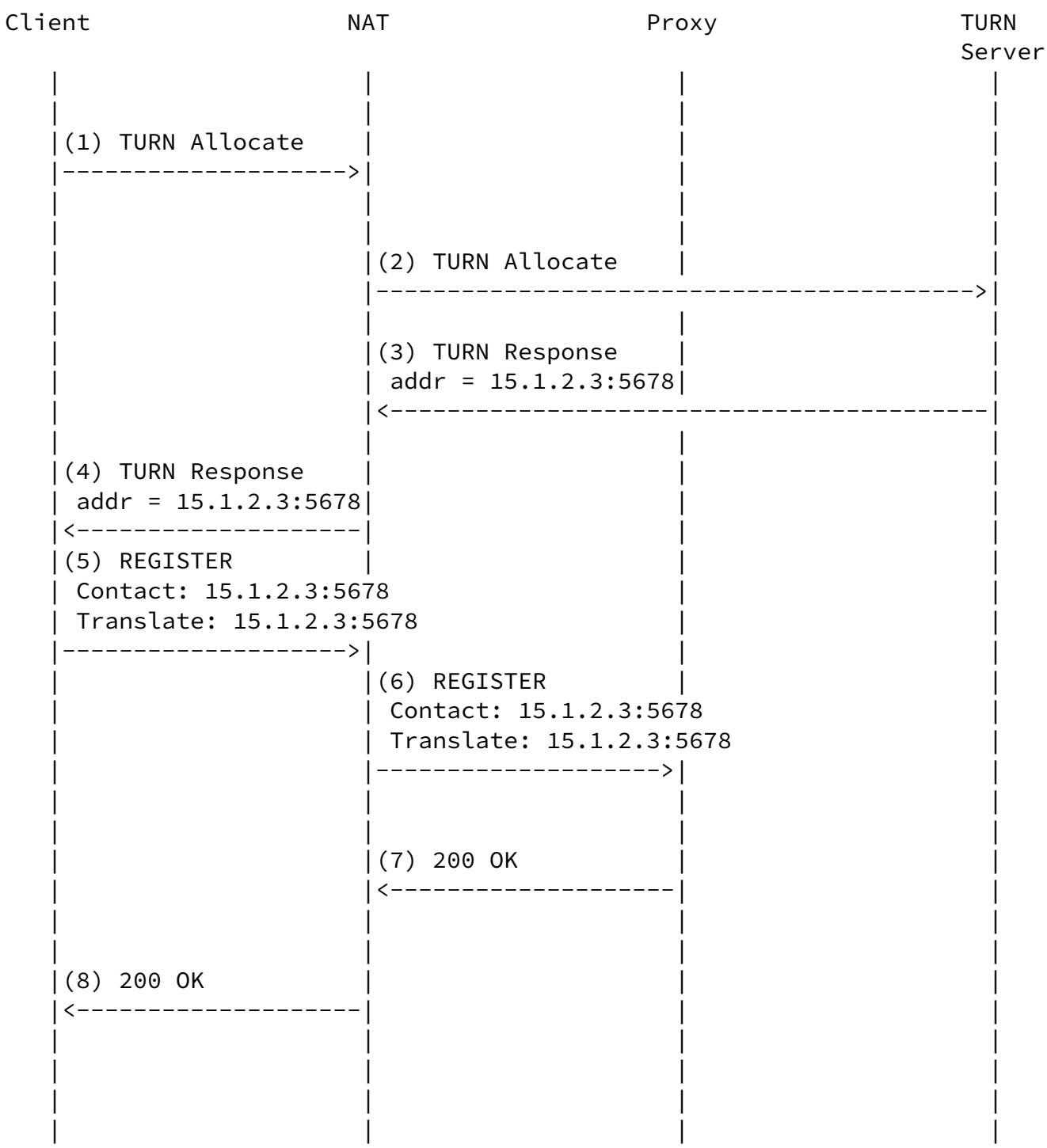
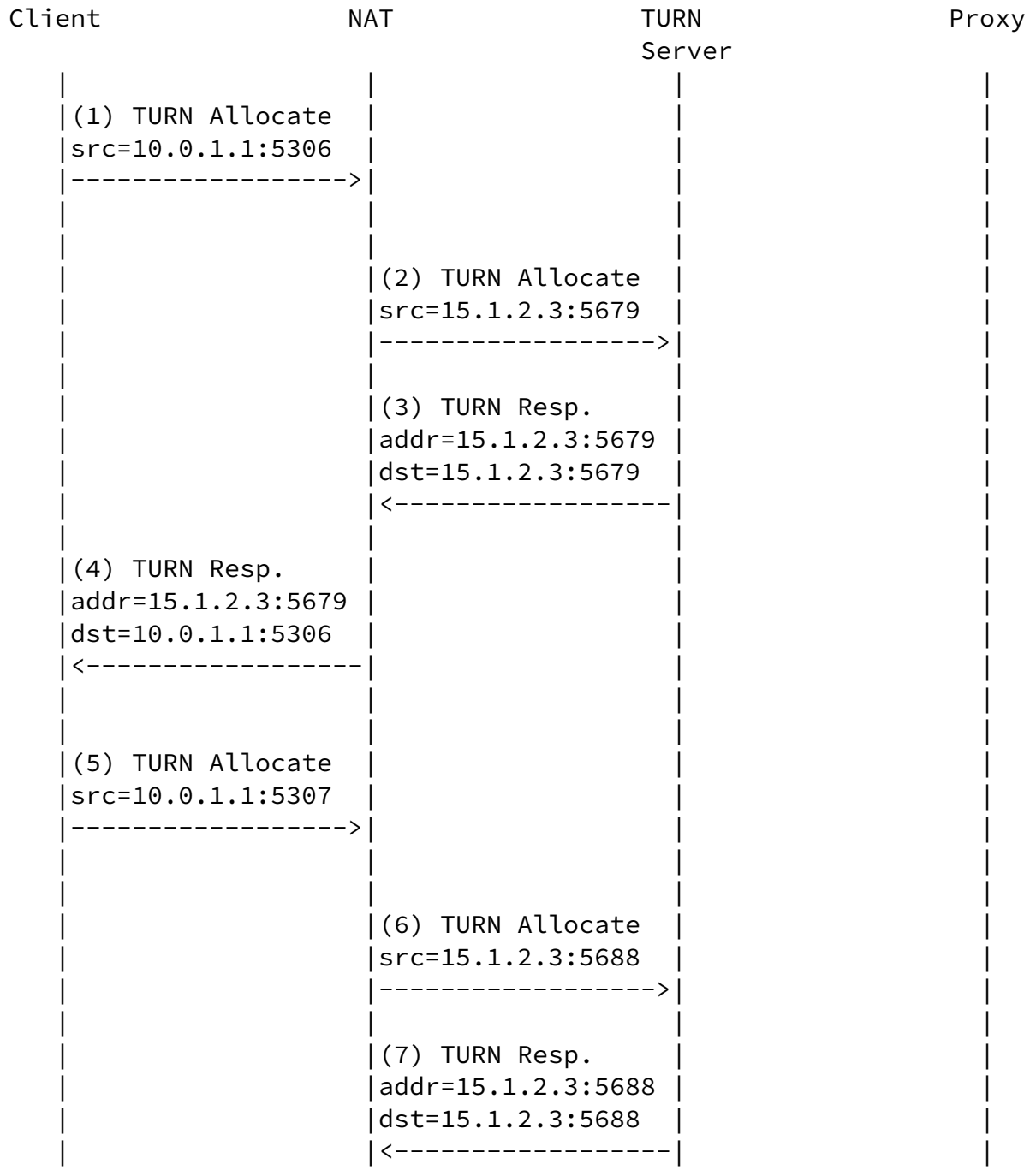


Figure 8: Registration for Symmetric NAT

```
   Contact: sip:15.1.2.3:5678
   Content-Type: application/sdp
   Content-Length: ...
```

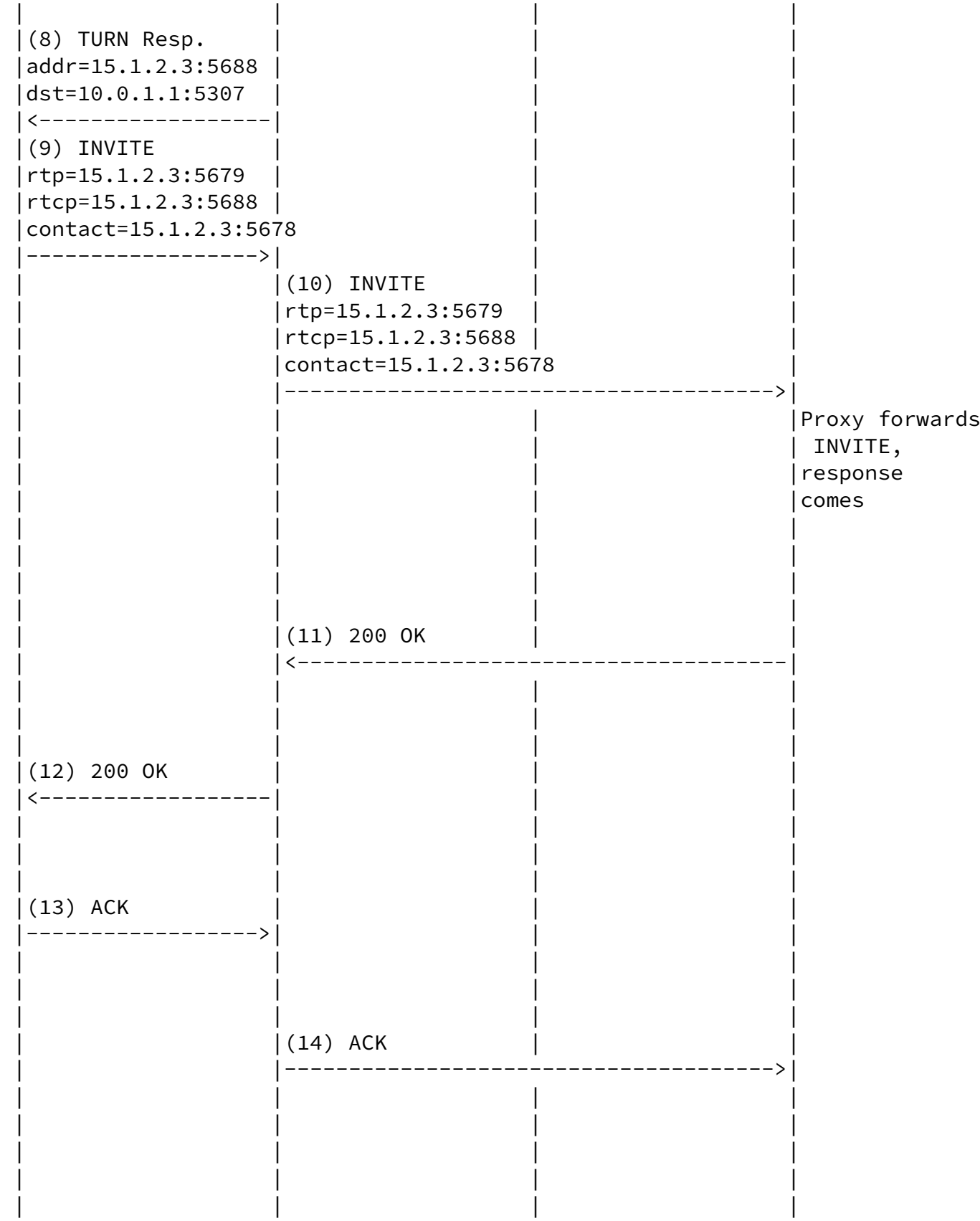


Figure 9: STUN Symmetric, Outgoing INVITE

```
v=0
o=aa 2890844526 2890842807 IN IP4 15.1.2.3
c=IN IP4 15.1.2.3
t=0 0
m=audio 5679 RTP/AVP 0
a=rtcp:5688
a=direction:active
```

2.2.2.3 Answering an Invitation to a Session

The procedure for answering an INVITE mirrors that for the non-symmetric case described in Section 2.2.1. However, the RTP and RTCP ports are obtained from a TURN server, not a STUN server, and the client should indicate a=direction:active in the response if the INVITE request indicated passive or both (assuming it supports symmetric RTP.)

Another difference is the path of the INVITE itself. If the proxy doesn't support the SIP extensions for NAT [9], the Translate header in the REGISTER will be ignored, and the address in the Contact header will be used. This address was obtained through the TURN server, so that the incoming INVITE will flow from the proxy, through the TURN server, towards the client.

The call flow for this case is shown in Figure 10. Note that the incoming INVITE, its 200 OK, and the ACK, now all flow through the TURN server. The initial INVITE from the proxy to the TURN server (message 1) will actually cause a TCP connection to be first opened from the proxy to the turn server. From this point forward, the client should re-register over this TCP connection (rather than the other one it used to send the initial REGISTER, which goes directly to the proxy), without the Translate header. Using this connection for re-registrations is needed to keep the connection alive.

The 200 OK response sent by the client might look like:

```
SIP/2.0 200 OK
From: sip:user@school.edu;tag=88asd
To: sip:user@domain;tag=99as8a
Call-ID: 98asd6asd60099
CSeq: 987769 INVITE
Via: SIP/2.0/TCP 1.9.2.2
```

```
Via: SIP/2.0/UDP 82.3.4.5
Via: SIP/2.0/TCP 15.1.2.3
```



```
Contact: sip:15.4.5.6:7766
Content-Type: application/sdp
Content-Length: ...

v=0
o=aa 2890844526 2890842807 IN IP4 15.4.5.6
c=IN IP4 15.4.5.6
t=0 0
m=audio 9988 RTP/AVP 0
a=rtcp:8877
a=direction:active
```

2.2.2.4 Media Flows

The media flow scenarios depend on the type of NAT each side is behind.

Consider the case where the caller is behind a full cone NAT, and the callee is behind a symmetric NAT. In this case, the SDP in the INVITE from the caller will have an a=direction:both attribute. The SDP in the 200 OK from the callee will have a=direction:active. The callee will send media to the caller, using a socket different than the one used to allocate the address from the TURN server. The caller sends media back to the source address where media from the callee came from. In this case, even though a TURN address was allocated by the callee, that address is never used, and the binding will expire at the TURN server.

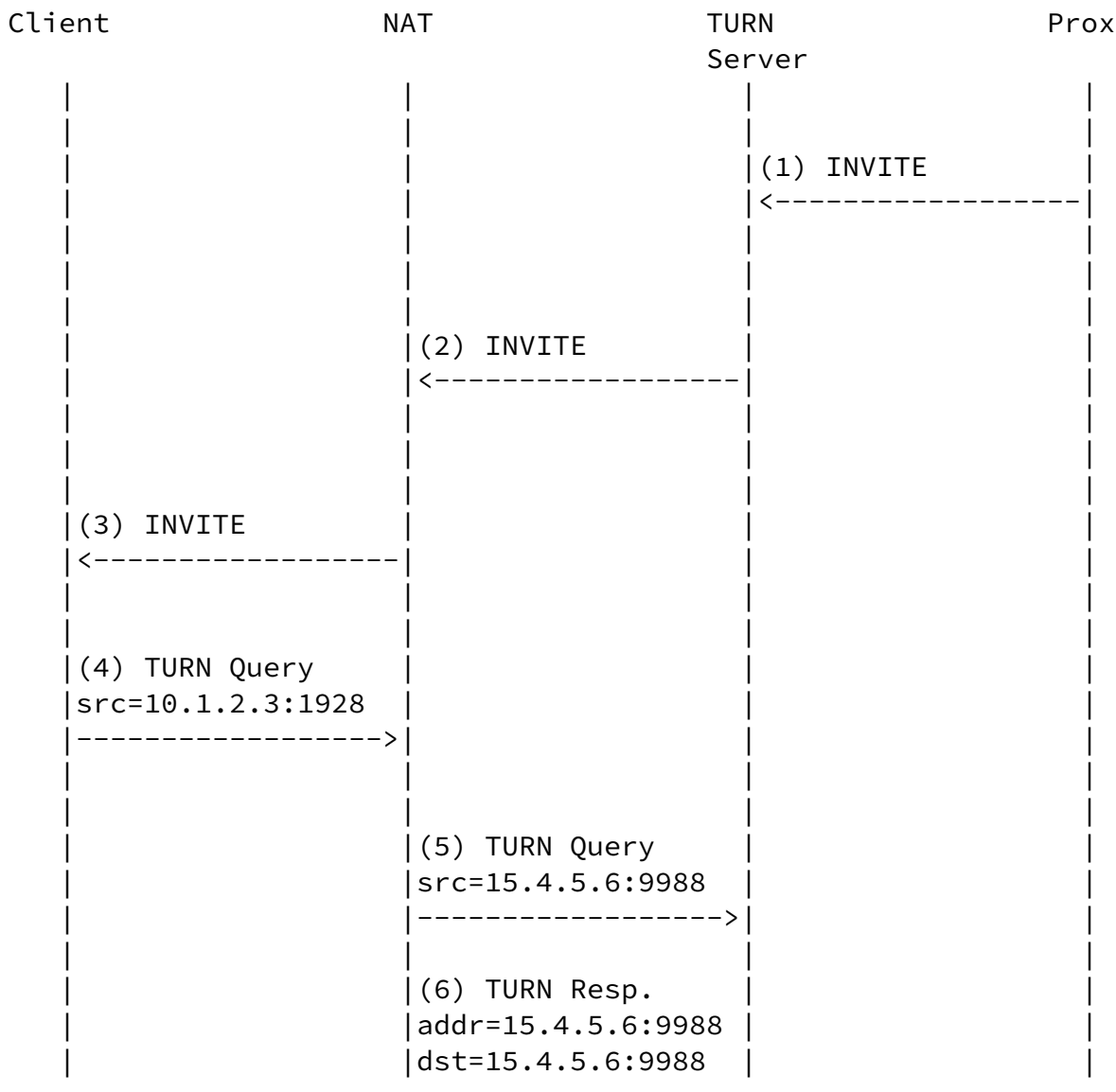
TODO: flow for this case

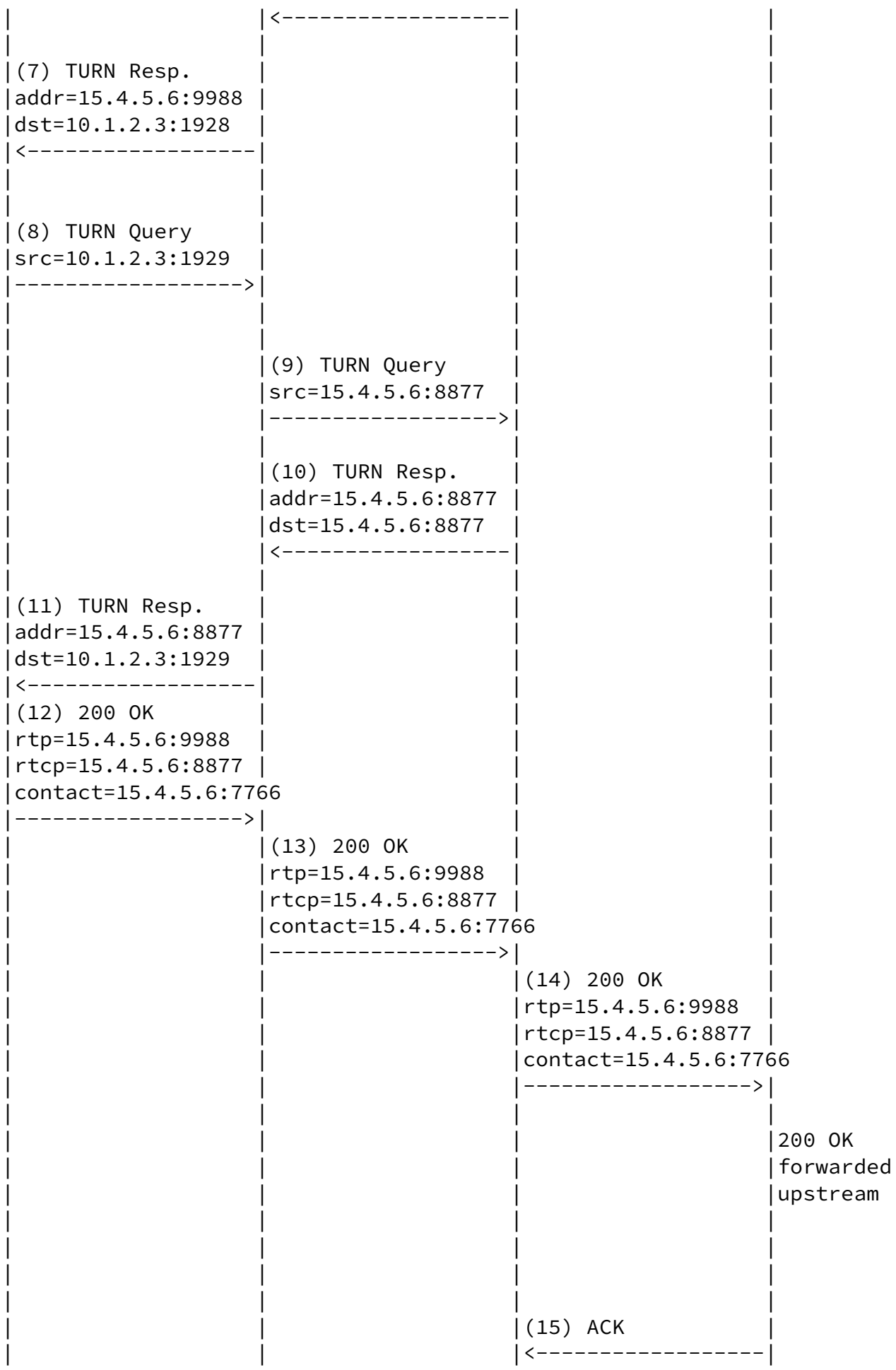
Consider the case where the caller is behind a restricted cone NAT, and the callee is behind a symmetric NAT. The caller includes an a=direction:active attribute in its INVITE. The callee responds with an a=direction:passive attribute. The caller will send to the address provided by the callee in the 200 OK. This address is on the TURN server. This data is received by the callee, which sends data back.

This is forwarded to the TURN server, and then onwards towards the caller. Thus, an intermediary is included in this case.

TODO: flow for this case

TODO: not sure all these cases are completely sorted out. Setting of active/passive for restricted cone requires more




(7) TURN Resp.
addr=15.4.5.6:9988
dst=10.1.2.3:1928
(8) TURN Query
src=10.1.2.3:1929
(9) TURN Query
src=15.4.5.6:8877
(10) TURN Resp.
addr=15.4.5.6:8877
dst=15.4.5.6:8877
(11) TURN Resp.
addr=15.4.5.6:8877
dst=10.1.2.3:1929
(12) 200 OK
rtp=15.4.5.6:9988
rtcp=15.4.5.6:8877
contact=15.4.5.6:7766
(13) 200 OK
rtp=15.4.5.6:9988
rtcp=15.4.5.6:8877
contact=15.4.5.6:7766
(14) 200 OK
rtp=15.4.5.6:9988
rtcp=15.4.5.6:8877
contact=15.4.5.6:7766
200 OK
forwarded
upstream
(15) ACK

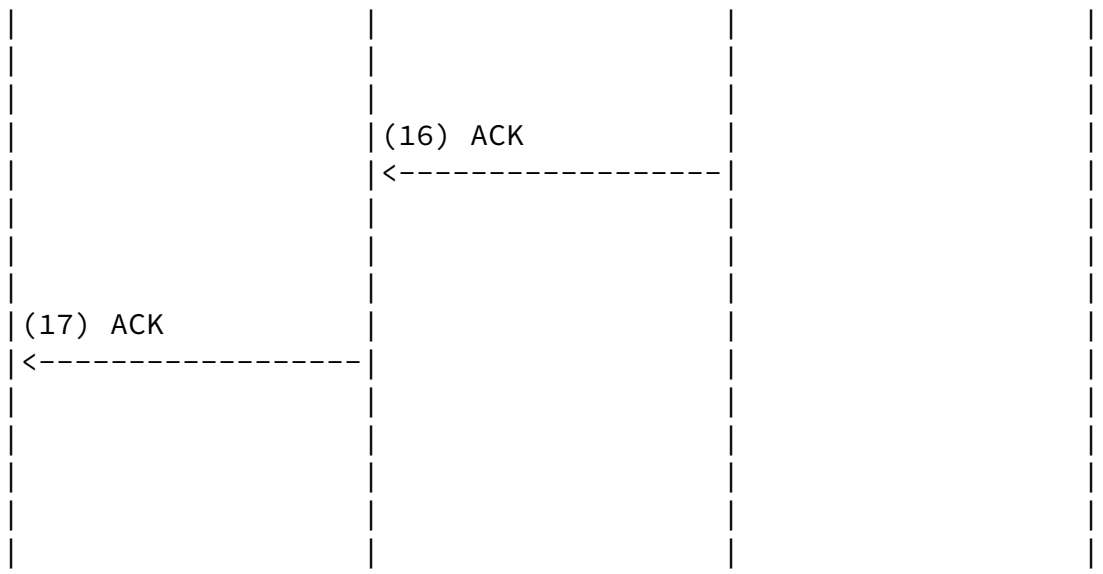


Figure 10: Incoming INVITE, Symmetric NAT

consideration.

2.3 Solution III: STUN in B2BUA

The solution of Section 2.2 assumed that the client would need to be upgraded to support at least STUN, and additionally several other SIP extensions. In many cases, this will not be feasible. It is possible to deploy the STUN solution in an intermediary, typically run as a piece of software that would be run within the residence. It can run on the same machine as the client, or on a separate PC. In SIP terminology, this additional server is a "B2BUA", or Back to back User Agent. It receives requests from the clients, acts as a STUN client, obtains addresses for media, modifies the SDP in the INVITEs, and forwards the requests onwards. Both SIP traffic, and media traffic, would be routed through it. Unlike the use of an intermediary in the service provider network (like a TURN server), the use of an intermediary in the home does not introduce any real latency or voice quality problems, and there is no cost to the service provider. There could be cost to the consumer, but we anticipate that these intermediaries would be given away for free as part of the service.

We call the B2BUA in the home network that acts as a STUN client a "STUN Agent", just for the sake of having a simple name. Note that

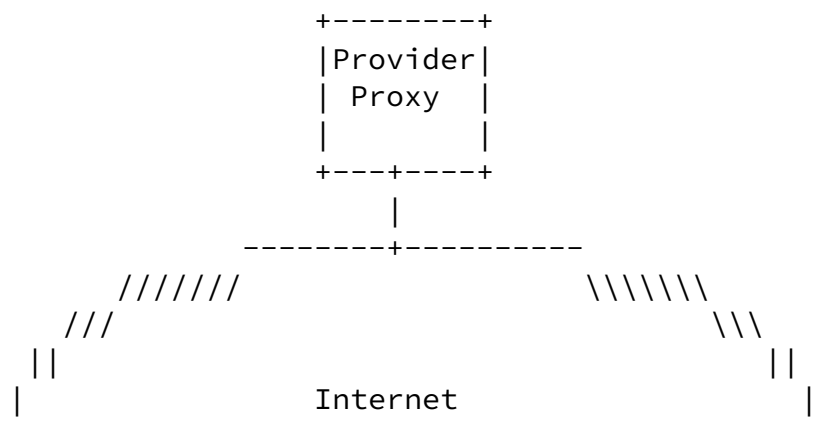
the STUN agent may also need to implement TURN, in the case that the agent is behind a symmetric NAT. However, in the flows below, we do specifically consider this case. The flows from Section 2.2.2 can be directly applied to the flows below to show this scenario.

The configuration using the STUN Agent is shown in Figure 11. The clients (PC phones, hardphones) are configured to talk to the STUN agent as their SIP local outbound proxy server, sending it both registrations and invitations.

When the STUN agent starts up, it discovers whether there is a NAT or not, and what type of NAT, using the STUN discovery mechanisms [2]. In the event that there is no NAT, the STUN agent merely acts as a stateless proxy, relaying packets in and out. We consider, from here on, the case where it has discovered that it is behind a full cone or restricted cone NAT.

2.3.1 Registration

The first step is for the clients to REGISTER. The flow for this scenario is shown in Figure 12. First, the client sends a REGISTER (message 1). This is a normal UDP REGISTER, and therefore the Contact



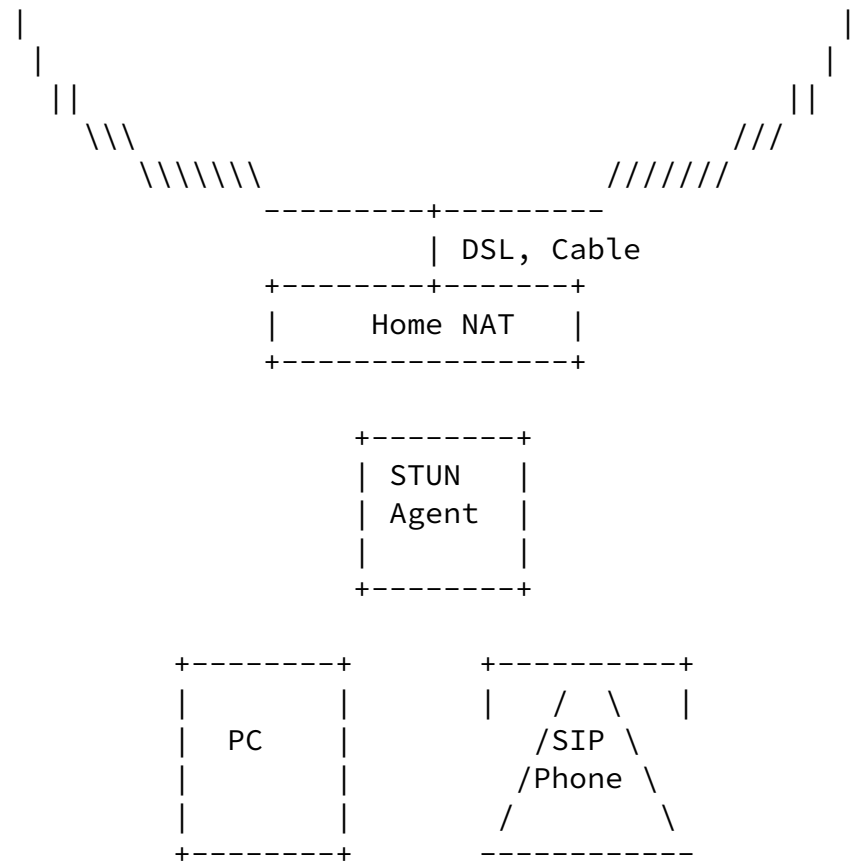
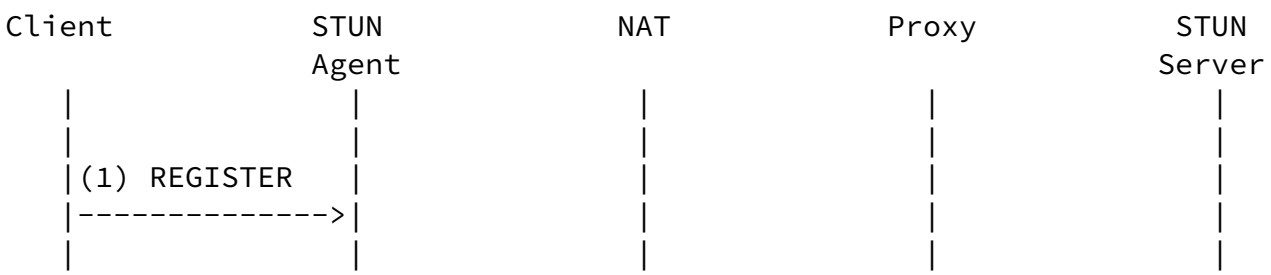


Figure 11: STUN Agent Configuration

is likely to reflect an IP address within the home network:



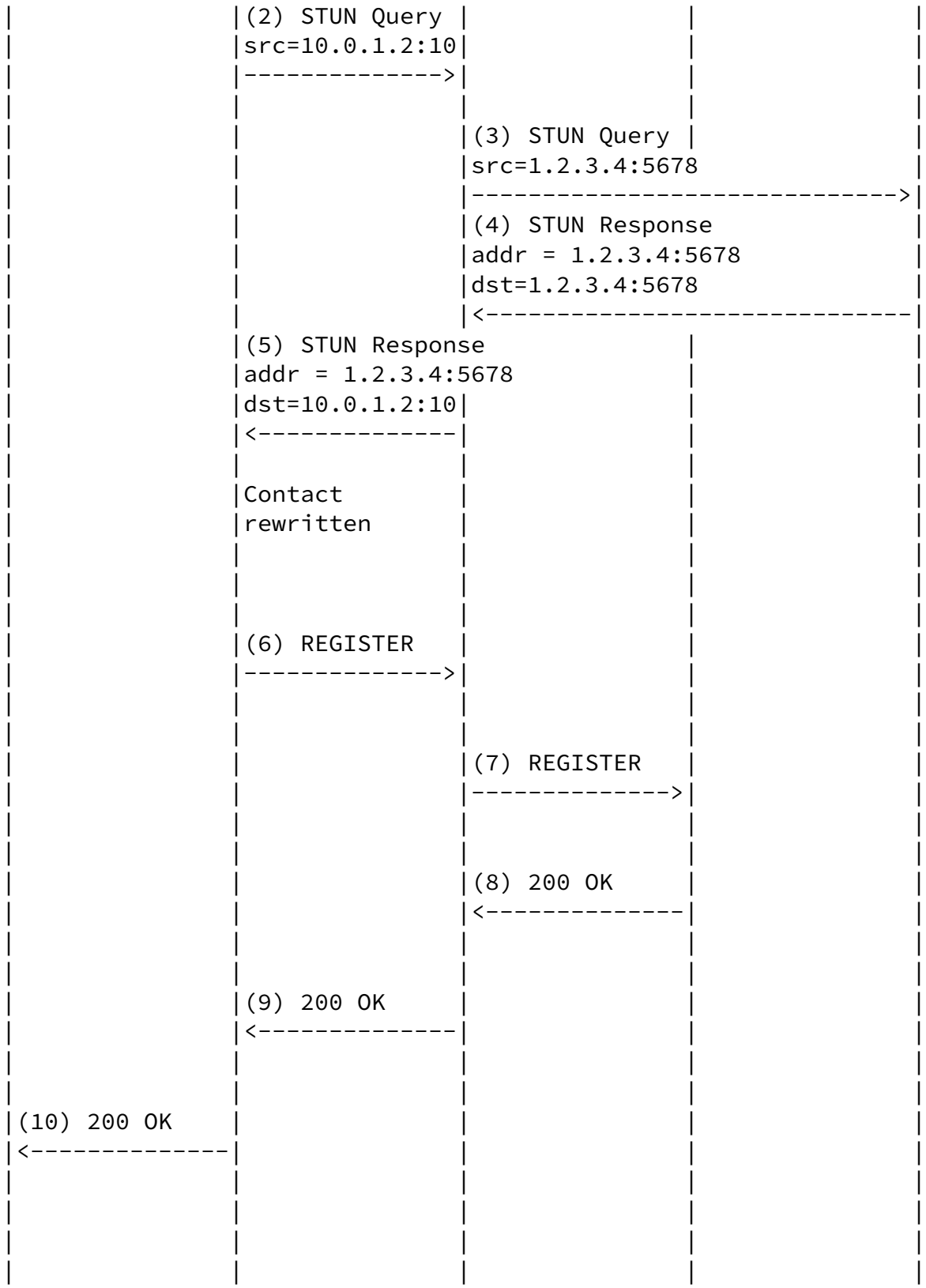


Figure 12: STUN Agent Registration

```
REGISTER sip:provider.com SIP/2.0
Via: SIP/2.0/UDP 10.0.1.1
From: sip:user@provider.com
To: sip:user@provider.com
Call-ID: 99asdasd98hnn
Contact: sip:user@10.0.1.1:5060
CSeq: 88790 REGISTER
```

Since the client is using a local outbound proxy, this REGISTER is sent to the STUN agent. The STUN agent recognizes that this is a registration, and determines that it needs to obtain an IP address and port to use in the Contact which can route to it. So, it follows the procedures that the client itself would have followed from Section 2.2.1. Specifically, it acts as a STUN client, and sends a Query request (message 2) to the provider.com STUN server. This goes through the NAT (message 3), and is reflected off of the STUN server (message 4) providing the STUN agent with a external IP address and port that routes to it (message 5). This address is inserted as the Contact header in a rewritten REGISTER message (message 6) that is sent to the proxy. This outgoing REGISTER might look like:

```
REGISTER sip:provider.com SIP/2.0
Via: SIP/2.0/UDP 1.2.3.4:5678
From: sip:user@provider.com
To: sip:user@provider.com
Call-ID: 88sdasdlasd0
Contact: sip:user@1.2.3.4:5678
Translate: sip:user@1.2.3.4:5678
CSeq: 88790 REGISTER
```

This REGISTER is forwarded through the NAT (message 7), and the 200 OK response from the proxy/registar (message 8) is sent through the NAT to the STUN agent (message 9). From this 200 OK, the STUN agent knows whether or not the Translate header was used or not.

Its important to note that the STUN query need only be done for the first REGISTER request. For subsequent ones, the STUN agent can reuse the same binding. The binding must be refreshed (if it was used), of course, but a new one need not be selected for each registration. The Contact header in each registration from a different client would have to contain a unique user name in order to disambiguate the incoming requests from the proxy.

The final step is for the STUN agent to respond to the REGISTER received from the client. The response contains the original Contact sent in the REGISTER. The STUN agent also remembers this contact, effectively acting as a registrar itself. This contact will be used for incoming INVITE requests. The response sent to the client might look like:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.0.1.1
From: sip:user@provider.com
To: sip:user@provider.com
Call-ID: 99asdasd98hnn
Contact: sip:user@10.0.1.1:5060
Expires:3600
CSeq: 88790 REGISTER
```

In this way, when the STUN agent receives an incoming INVITE with a request URI of sip:user@1.2.3.4:5678, it knows to rewrite it to sip:user@10.0.1.1:5060 and proxy the request there.

2.3.2 Initiating a Session

The call flow for initiating a session is shown in Figure 13. The client formulates a normal INVITE (message 1), without SIP extensions for NAT [9] or SDP extensions for RTCP [8]. It it is sent to the STUN Agent as a local outbound proxy. This INVITE might look like:

```
INVITE sip:user@school.edu SIP/2.0
Via: SIP/2.0/UDP 10.0.1.1
From: sip:user@provider.com
To: sip:user@school.edu
Call-ID: 99asdas88shnn
Contact: sip:user@10.0.1.1:5060
Content-Type: application/sdp
Content-Length: ...
```

```
v=0
o=aa 2890844526 2890842807 IN IP4 10.0.1.1
c=IN IP4 10.0.1.1
t=0 0
m=audio 9988 RTP/AVP 0
```


The STUN Agent needs to do several things. First, it must rewrite the Contact header to reference the signaling address bound to sip:user@10.0.1.1:5060 from the previous registration process (in this case, sip:user@1.2.3.4:5678). Secondly, it rewrites the SDP to include addresses learned from STUN, in much the same way the client learned them in Section 2.2.1 (messages 2-5 and message 6-9). It also adds its own Via address to the outgoing INVITE (message 10), which might look like:

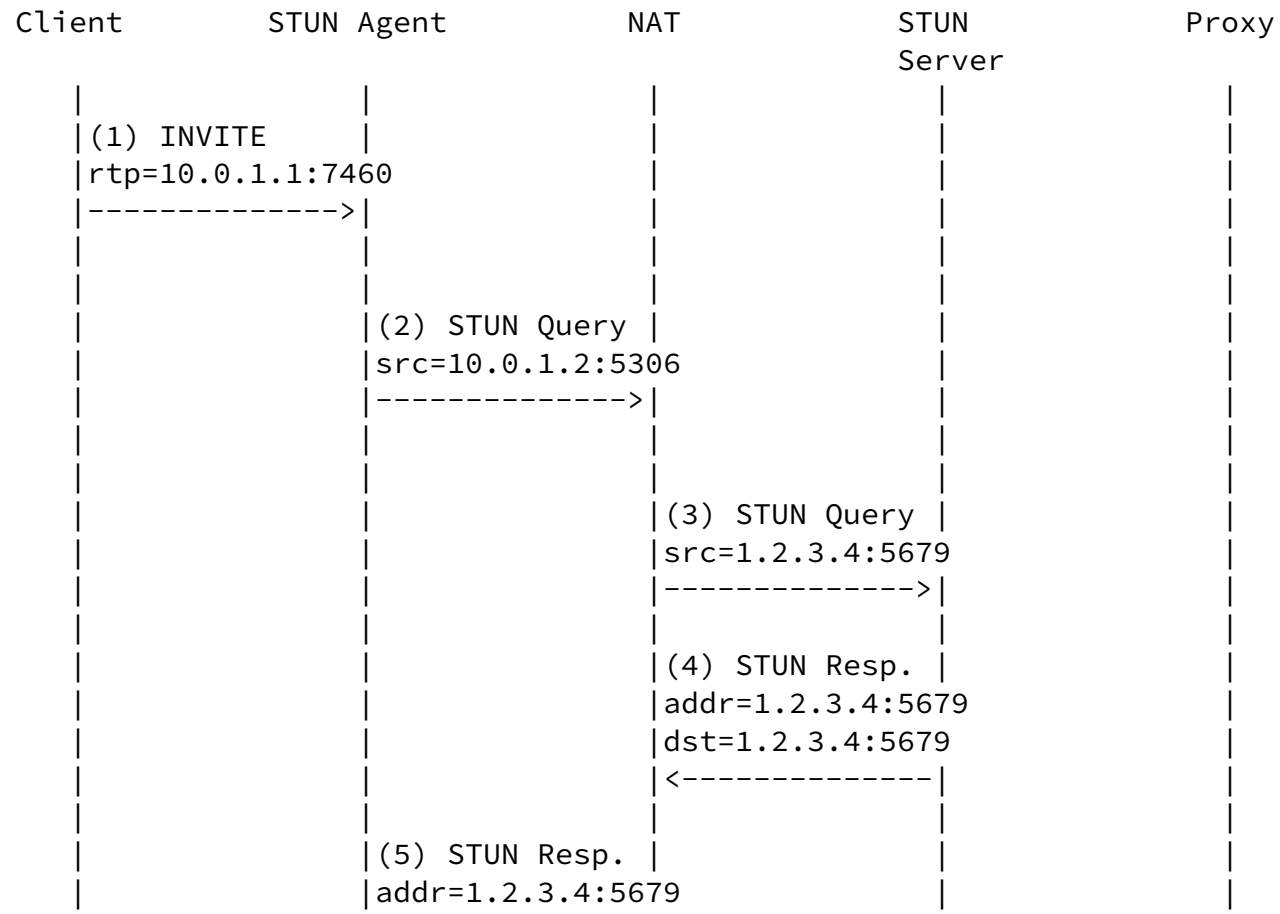
```
INVITE sip:user@school.edu SIP/2.0
Via: SIP/2.0/TCP 1.2.3.4:5678
Via: SIP/2.0/UDP 10.0.1.1
From: sip:user@provider.com
To: sip:user@school.edu
Call-ID: 99asdas88shnn
Contact: sip:user@1.2.3.4
Content-Type: application/sdp
Content-Length: ...

v=0
o=aa 2890844526 2890842807 IN IP4 1.2.3.4
c=IN IP4 1.2.3.4
t=0 0
m=audio 5679 RTP/AVP 0
a=rtcp:5688
a=direction:both
```

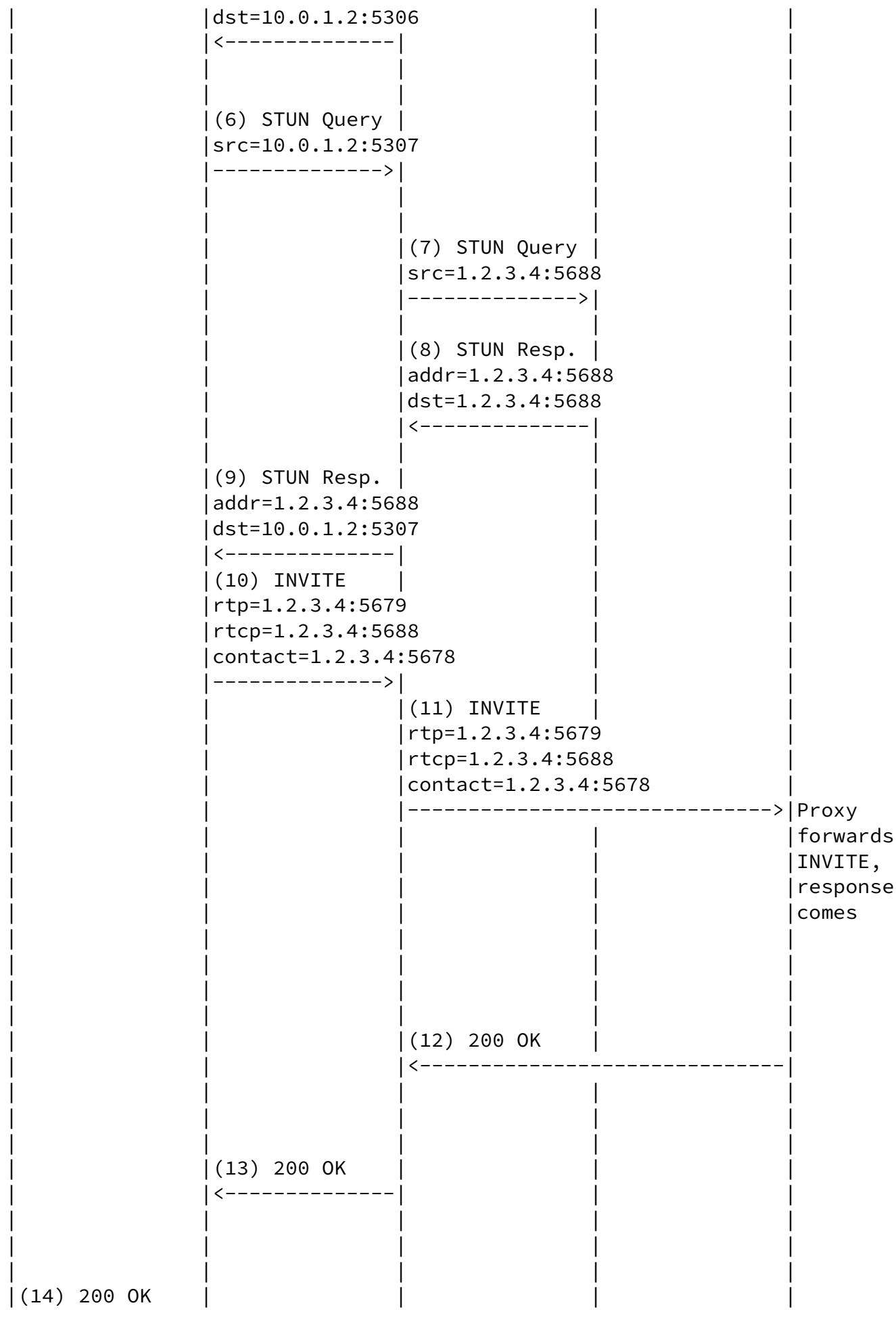
> There is a subtle error here, in that 1.2.3.4:5678 is the UDP address learnt from STUN. This can't be used in the Via

header sent over a TCP connection.

The 200 OK to this INVITE will be received by the STUN Agent (message 13). This 200 OK need not be rewritten if it does not contain a direction or rtcp attribute. In that case, media flows out from the client directly towards the address in the 2xx, and media flows in first through the intermediary, and then towards the client. Let us consider the case where the 200 OK does include an a=rtcp attribute:

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP 1.2.3.4:5678
Via: SIP/2.0/UDP 10.0.1.1
From: sip:user@provider.com
To: sip:user@school.edu;tag=909asd
```


dst=10.0.1.2:5306
(6) STUN Query
src=10.0.1.2:5307
(7) STUN Query
src=1.2.3.4:5688
(8) STUN Resp.
addr=1.2.3.4:5688
dst=1.2.3.4:5688
(9) STUN Resp.
addr=1.2.3.4:5688
dst=10.0.1.2:5307
(10) INVITE
rtp=1.2.3.4:5679
rtcp=1.2.3.4:5688
contact=1.2.3.4:5678
(11) INVITE
rtp=1.2.3.4:5679
rtcp=1.2.3.4:5688
contact=1.2.3.4:5678
Proxy forwards INVITE, response comes
(12) 200 OK
(13) 200 OK
(14) 200 OK

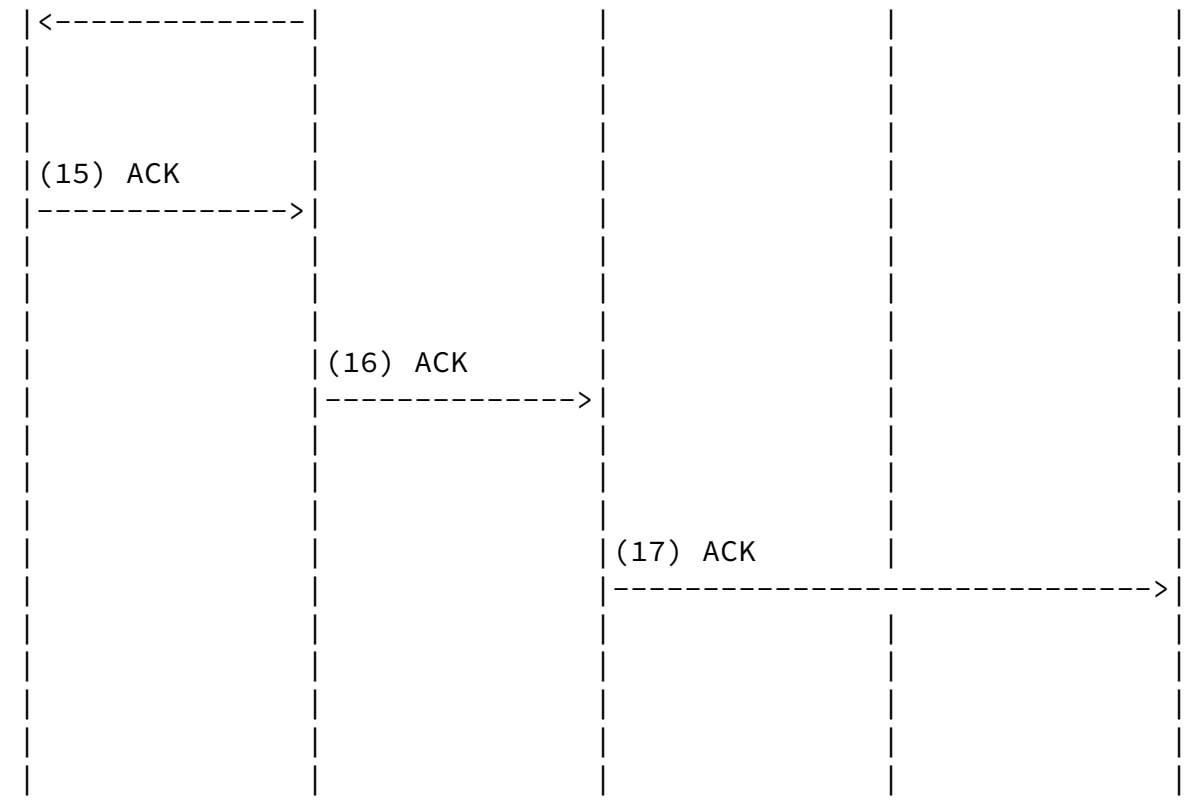


Figure 13: Initiating a session with a STUN Agent


```
Call-ID: 99asdas88shnn
Contact: sip:user@56.66.77.88
Content-Type: application/sdp
Content-Length: ...

v=0
o=aa 2890844526 2890842807 IN 56.66.77.88
c=IN IP4 56.66.77.88
t=0 0
m=audio 56496 RTP/AVP 0
a=rtcp:36485
```

Since the client doesn't support the RTCP extensions for SDP, the STUN Agent needs to allocate a port pair on the host which it can rewrite into this 200 OK before forwarding. No STUN requests are needed to allocate this port pair; its local on the STUN Agent. The

forwarded 200 OK might look like:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.0.1.1
From: sip:user@provider.com
To: sip:user@school.edu;tag=909asd
Call-ID: 99asdas88shnn
Contact: sip:user@56.66.77.88
Content-Type: application/sdp
Content-Length: ...

v=0
o=aa 2890844526 2890842807 IN 56.66.77.88
c=IN IP4 10.0.1.2
t=0 0
m=audio 4372 RTP/AVP 0
```

The media flow for this case is shown in Figure 14. As the flow shows, both RTP and RTCP in and out flow through the STUN Agent. In the case of outgoing RTP and RTCP, the media is sent on the sockets used to send the STUN queries, and addressed to the addresses and ports from the 200 OK. For incoming RTP and RTCP, its received on the sockets used for the stun queries, and from there, forwarded out to the address in the original INVITE from the client. Notice how the STUN Agent can guarantee that media sent to and from the client has adjacent RTP and RTCP ports.

2.3.3 Receiving an Invitation to a Session

The call flow for handling an incoming INVITE is shown in Figure 15. As expected, this is a mirror image of the outgoing INVITE case 13. We have simplified this flow, however, and assumed that the incoming INVITE (message 2) did not require the usage of SDP attributes for RTCP or connection oriented media. As a result, the INVITE can be passed directly to the client (message 3). The 200 OK returned by the client (message 4) contains an SDP media address that needs to be rewritten by the agent.

Media from the caller to the callee will flow first through the STUN Agent. From there it is forwarded to the callee (using adjacent RTP/RTCP ports on this leg). However, media from the callee flows directly to the caller at the address specified in the incoming INVITE.

2.4 Solution IV: ALG

Another solution for the residential case is to throw away your NAT, and buy one with an embedded ALG inside. This will allow SIP to work transparently through it without any additional infrastructure on the service provider or residence side. It will not require any changes to the clients either.

The primary drawback to this approach is its impracticality. At the time of writing, there is only one residential NAT product available with SIP support. This product is not from a maintstream vendor, and at the time of writing, it is still missing features that are generally baseline requirements for such a device. Whilst this will certainly be corrected in a future product release, its symptomatic of the bigger problem - the manufacturer of a residential NAT cannot, and won't, be the expert in all application protocols that would need ALG support. Similarly, a vendor that specializes in a NAT supporting a particular ALG, won't be the expert in other general purpose NAT features, or in other protocols.

Protocols also evolve, so that even if SIP support were added today, its not clear that this would support the extensions being developed tomorrow.

A further problem is that there is a large, installed based of residential NATs that are not SIP aware. It is unlikely that users would be willing to pay money to buy a new box to support a particular application. Most users would expect that the provider would simply "make it work". Asking the user to purchase a new, SIP enabled ALG is putting the responsibility for making it work into the

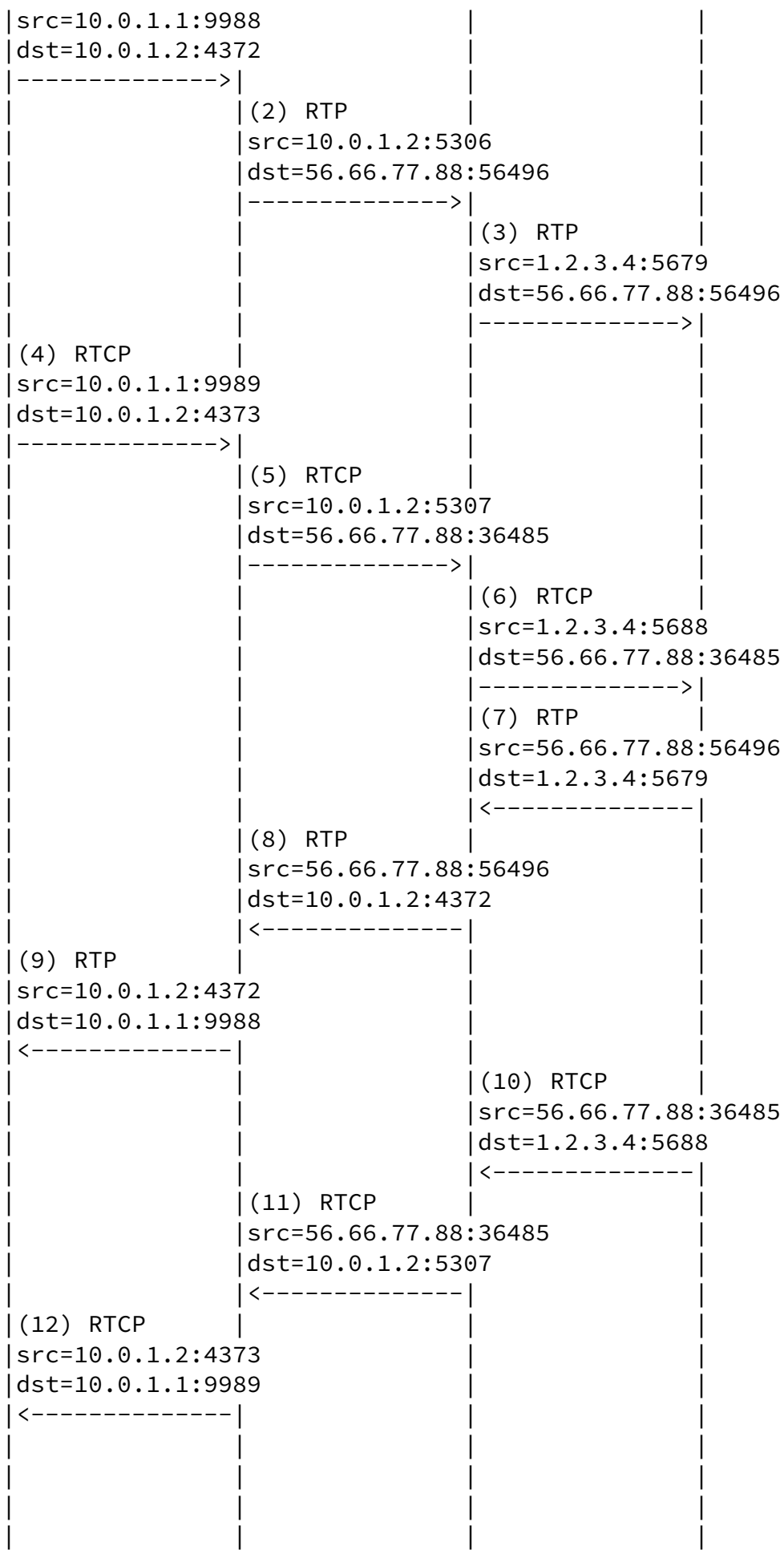
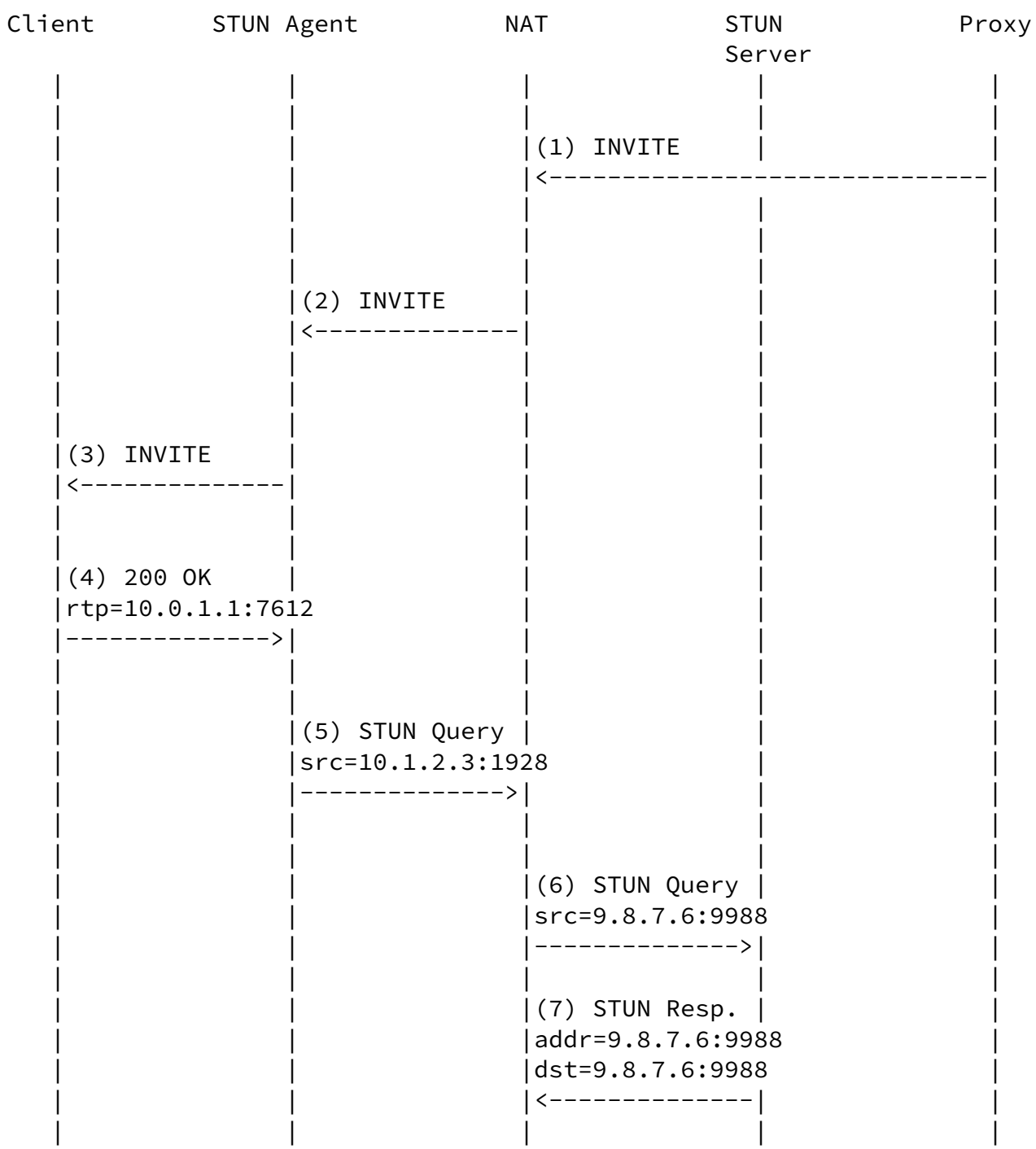
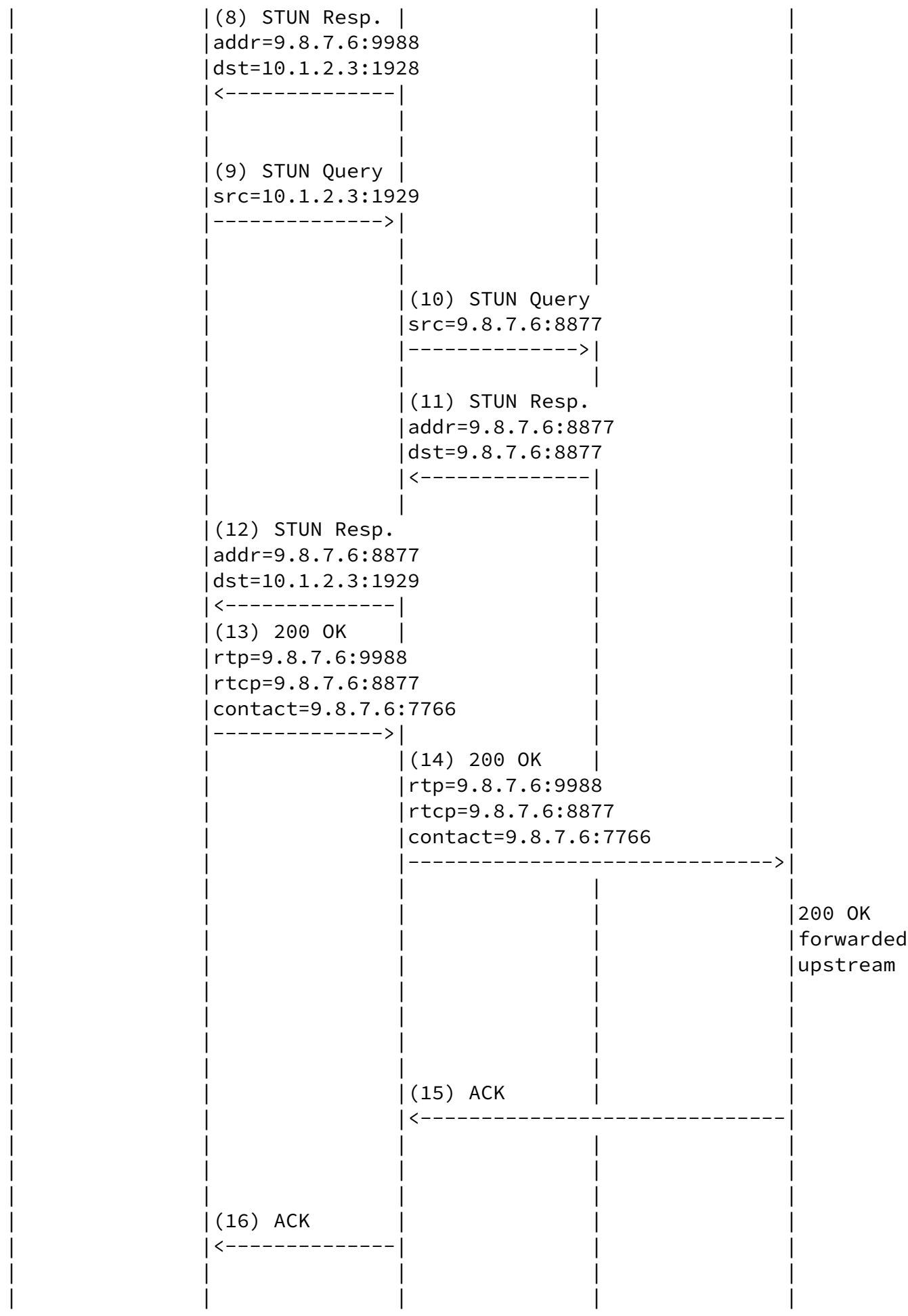

src=10.0.1.1:9988
dst=10.0.1.2:4372
(2) RTP
src=10.0.1.2:5306
dst=56.66.77.88:56496
(3) RTP
src=1.2.3.4:5679
dst=56.66.77.88:56496
(4) RTCP
src=10.0.1.1:9989
dst=10.0.1.2:4373
(5) RTCP
src=10.0.1.2:5307
dst=56.66.77.88:36485
(6) RTCP
src=1.2.3.4:5688
dst=56.66.77.88:36485
(7) RTP
src=56.66.77.88:56496
dst=1.2.3.4:5679
(8) RTP
src=56.66.77.88:56496
dst=10.0.1.2:4372
(9) RTP
src=10.0.1.2:4372
dst=10.0.1.1:9988
(10) RTCP
src=56.66.77.88:36485
dst=1.2.3.4:5688
(11) RTCP
src=56.66.77.88:36485
dst=10.0.1.2:5307
(12) RTCP
src=10.0.1.2:4373
dst=10.0.1.1:9989

Figure 14: Media flow for STUN Agent




(8) STUN Resp.
addr=9.8.7.6:9988
dst=10.1.2.3:1928
(9) STUN Query
src=10.1.2.3:1929
(10) STUN Query
src=9.8.7.6:8877
(11) STUN Resp.
addr=9.8.7.6:8877
dst=9.8.7.6:8877
(12) STUN Resp.
addr=9.8.7.6:8877
dst=10.1.2.3:1929
(13) 200 OK
rtp=9.8.7.6:9988
rtcp=9.8.7.6:8877
contact=9.8.7.6:7766
(14) 200 OK
rtp=9.8.7.6:9988
rtcp=9.8.7.6:8877
contact=9.8.7.6:7766
200 OK
forwarded
upstream
(15) ACK
(16) ACK

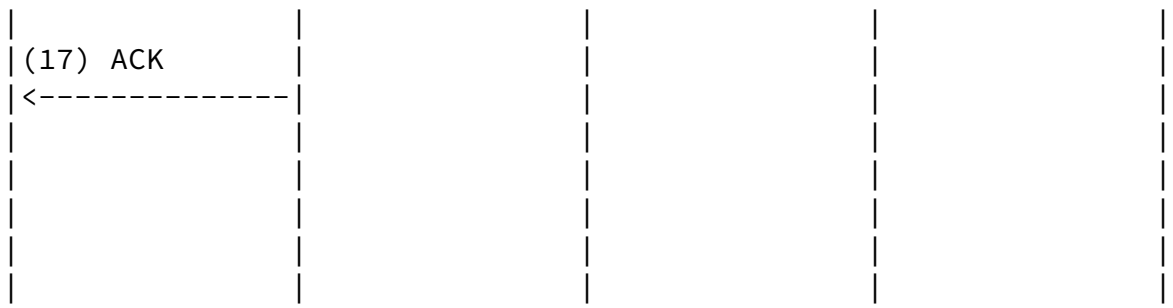


Figure 15: Incoming INVITE, STUN Agent

wrong hands. This is coupled with the fact that consumers are generally ignorant of technology, and would not be able to identify and purchase a NAT that had SIP "ALG" support in it. Certainly, most providers in existence today are not willing to wait for this massive upgrade to happen. They need a way to provide service now. Providers are motivated to change or upgrade equipment that they control (their own servers, the client software, hardphones) in order to provide service, but users are not generally motivated to change equipment for a new application. Therefore, we believe that waiting for SIP ALG support in residential NATs is an untenable proposition from a business perspective.

3 Uncooperative Enterprise

The "uncooperative enterprise" case arises when there are users within an enterprise that wish to access service from a provider on the public Internet. However, the enterprise uses a firewall or NAT. The enterprise has not deployed VoIP, and has not added any explicit configuration to support SIP, nor is there desire to do so.

This case is actually identical to the residential case for the most part. Nearly all of the solutions for that environment can be used here, but there are some differences.

First, the configuration trick of Section 2.1 does not work. That trick required that the user of the VoIP application had administrative control over the NAT. This is true in the residential case, but not in the enterprise case.

Secondly, enterprise NATs often include firewall functionality and

are more restrictive than the general NAT. At the extreme, they may even block all UDP traffic in and out. In these cases, there are few elegant or good solutions. The only workable approaches are the tunneling and VPN solutions. However, these are risky since they can introduce security weaknesses into the enterprise, and go against the enterprise policy that the firewall/NAT is trying to enforce in the first place.

If the enterprise firewall/NAT is not any more restrictive than the typical residential NAT (will allow outgoing UDP and TCP), the residential solutions all work fine.

4 Cooperative Enterprise

In this scenario, there is an enterprise which wishes to deploy SIP

services. There is no public provider involved at all; the enterprise takes full control. In this case, the enterprise firewall administrators do have the ability to manipulate the configuration of the firewall/NAT if neeeded, or to add and deploy additional elements inside of the network.

4.1 Solution I: ALG

With this solution, the enterprise upgrades their firewall, or purchases a new firewall, to run software with an integrated SIP ALG. The NAT/Firewall examines each SIP request, and modifies the Contact, session description, and Vias as appropriate. Note that this approach does not work if the SIP messages are hop-by-hop encrypted (using for example TLS) unless the NAT/Firewall also acts as a SIP proxy server.

Call flows for SIP ALGs have been documented in [6].

In addition, the enterprise deploys proxies internal to the enterprise. The NAT is configured with a static mapping or "conduit" for each of these internal SIP servers from public IP addresses (typically on the well-know SIP port--5060) to their internal IP addresses. The Enterprise can then advertise these public addresses in DNS A or SRV records. This configuration is shown in Figure 16.

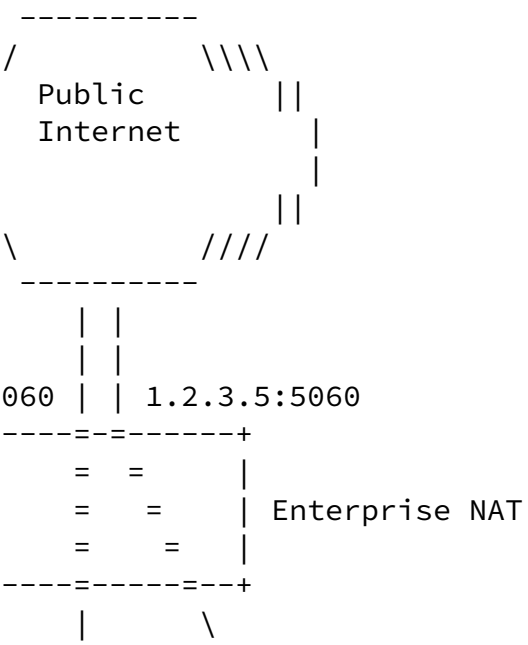
Registration is performed entirely inside the enterprise network. SIP

UAs inside the network can communicate with each other directly. SIP UAs wishing to communicate with the outside world need to go through the enterprise SIP proxy.

When initiating a session outside the enterprise, the NAT/FW rewrites portions of the SIP INVITE and offered session description to contain public IP addresses. The NAT/Firewall also assigns a pairs of public ports for RTP and RTCP as needed, maps these to the private addresses and ports included in the original session description, and allows incoming traffic to these ports from the public internet.

For example, the following INVITE might be rewritten as shown in the next example.

```
INVITE sip:user@domain SIP/2.0
From: sip:user@work.com;tag=88asd
To: sip:user@domain
Call-ID: 98asd6asd60099
CSeq: 987769 INVITE
Via: SIP/2.0/UDP 10.1.1.5
Via: SIP/2.0/UDP proxy1.work.com;maddr=10.1.1.1
```

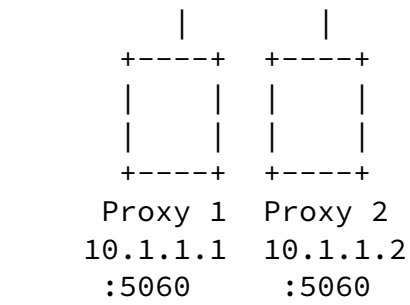


Figure 16: Enterprise ALG Configuration

```
Record-Route: proxy1.work.com
Contact: sip:10.1.1.5:5060
Content-Type: application/sdp
Content-Length: ...

v=0
o=aa 2890844526 2890842807 IN IP4 10.1.1.5
c=IN IP4 10.1.1.5
t=0 0
m=audio 17832 RTP/AVP 0
```

The same example INVITE shown rewritten.


```
INVITE sip:user@domain SIP/2.0
From: sip:user@work.com;tag=88asd
To: sip:user@domain
Call-ID: 98asd6asd60099
CSeq: 987769 INVITE
Via: SIP/2.0/UDP 10.1.1.5
Via: SIP/2.0/UDP proxy1.work.com;maddr=1.2.3.4:5060
Record-Route: proxy1.work.com
Contact: sip:1.2.3.6:7843
Content-Type: application/sdp
Content-Length: ...
```

```
v=0
o=aa 2890844526 2890842807 IN IP4 10.1.1.5
c=IN IP4 1.2.3.4
t=0 0
m=audio 5678 RTP/AVP 0
```

Likwise, when receiving sessions, the NAT/FW rewrites the answered session description. Note that the NAT/FW should be prepared to rewrite an offer in a 200, and an answer in an ACK.

4.2 Solution II: MIDCOM Firewall and Proxy

In this approach, the enterprise purchases a firewall/NAT which uses some type of firewall control protocol such as the protocol being developed in the MIDCOM Working Group. A SIP proxy inside the network performs the stateful inspection and ALG rewriting logic that was imbedded in the firewall in scenario 5.1. The proxy then opens and closes pinholes in the firewall or creates and destroys NAT mappings as needed to allow SIP and session media to flow through the middlebox. This approach is being discussed in depth in the MIDCOM WG, so we will not elaborate here.

4.3 Solution III: Internal B2BUAWM

In this solution, the enterprise does not need to change their NAT or firewall to get one with ALG support. Instead, they add an additional element inside the enterprise, which we call a "Back to Back User Agent With Media" or B2BUAWM. This element terminates all media and all SIP traffic in and out of the enterprise. As a result, the administrator of the firewall can configure the firewall to allow all traffic in and out from the B2BUAWM. Similarly, if NAT is used, a static mapping of a single public address to the private address of the B2BUAWM can be configured (i.e., using basic NAT, not NAPT), or, a range of ports can be allocated to the B2BUAWM. The media relay

component of the B2BUAWM can also be physically separated, using a control protocol, like MGCP [16] or MEGACO [17], between them.

Effectively, the B2BUAWM acts as a "DMZ host" for the SIP and media

traffic.

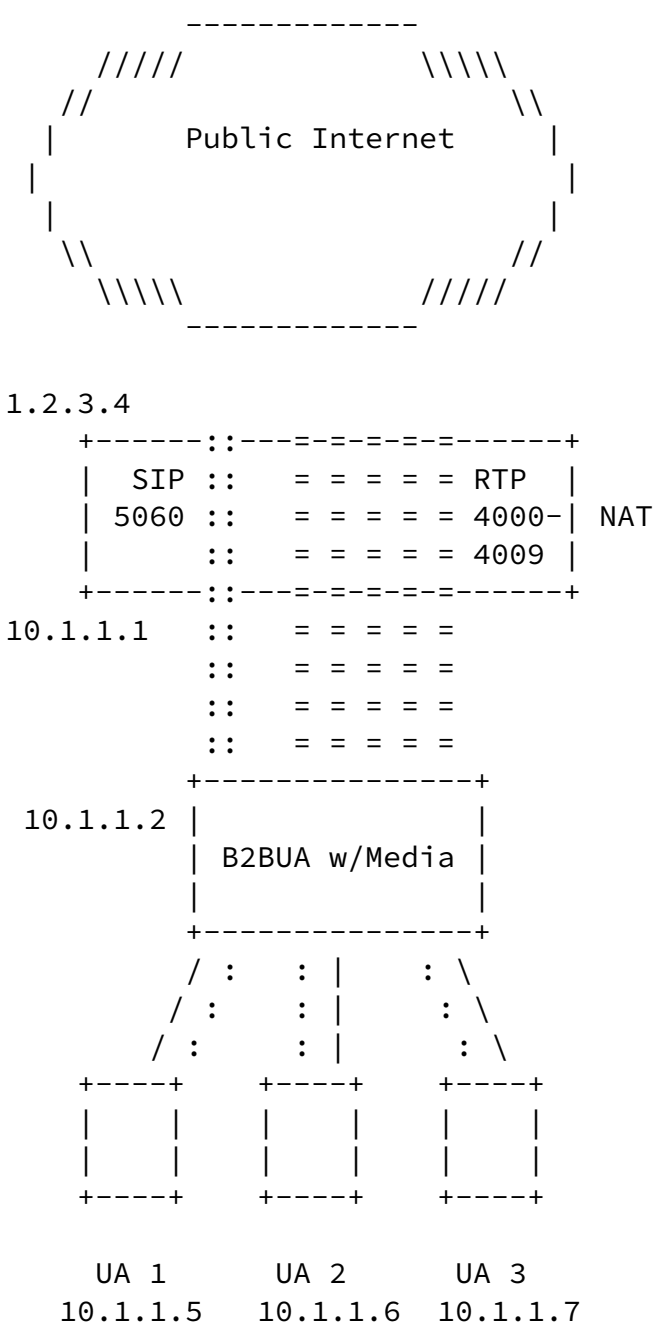


Figure 17: B2BUAWM for the Enterprise

In the example of Figure 17, an enterprise NAT is configured to map ports 4000 through 4009 from 1.2.3.4 (the public address of the NAT) to 10.1.1.2 (the private address of the media relay). The NAT is likewise configured to forward public SIP traffic on port 5060 to the B2BUA.

The B2BUA will rewrite all Contact and Via headers to appear as if they are coming from the public or private address of the B2BUA. This process is pretty much identical to the process that a SIP ALG inside of a NAT would use.

The B2BUA will allocate pairs of RTP/RTCP ports from its statically mapped pool, and rewrite the session description and each message, so that private addresses are translated to public addresses and vice versa. Figure 18 shows the call flow for an outgoing INVITE, and Figure 19 shows the call flow for an incoming INVITE.

5 Outsourced Service: Centrex

In the outsourced service scenario (also known as Centrex), the enterprise wishes to deploy the SIP service, but does so by outsourcing it to a provider in the public Internet.

5.1 Solution I: ALG

In this solution, the enterprise upgrades their firewall/NAT to include SIP ALG support. Unlike the cooperative enterprise case above, there is no need to deploy any proxies or additional elements inside the network (that is the point of Centrex). Clients within the enterprise register directly with the proxies owned and provided by the Centrex provider. This means that the ALG needs to rewrite REGISTER messages, in addition to INVITE/200/ACK messages. This is in contrast to an ALG deployed to support a cooperative enterprise. In that scenario, registrations are confined to the enterprise and never traverse the firewall/NAT.

This configuration is shown in Figure 20. The enterprise only has clients. These clients are configured with the IP address of the proxy in the provider's network that they will be using. The enterprise configures its DNS so that SRV lookups on its domain names for SIP service (i.e., _sip._udp.enterprise.com) route to the provider's proxies.

A call flow for a registration, followed by an incoming call, is

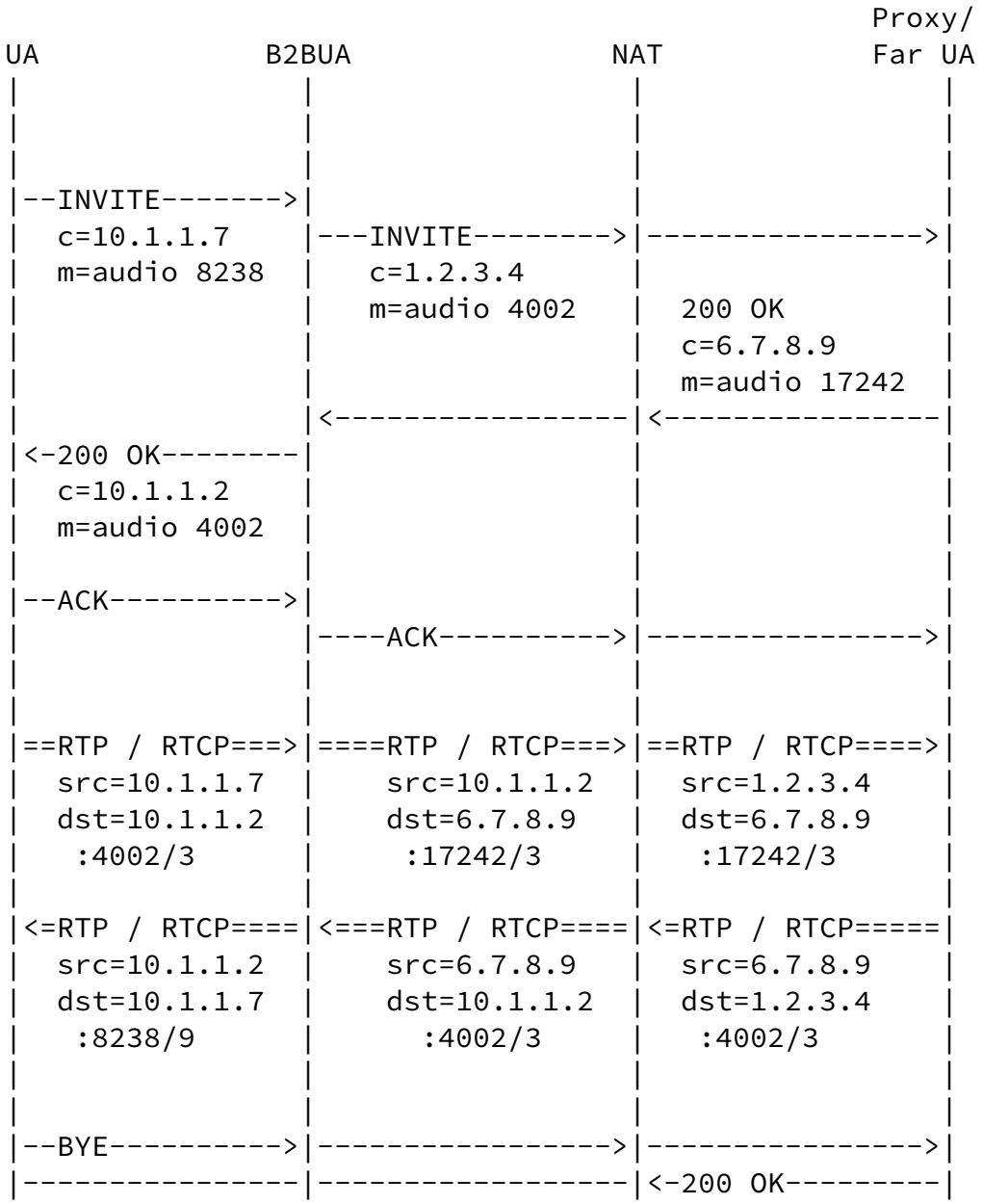


Figure 18: Enterprise with B2BUAWM: Outgoing Invite

shown in Figure 21.

When the clients start up, they send REGISTER requests to this proxy (message 1). The ALG is responsible for rewriting the Contact in the registrations. The bindings allocated for the Contact headers are set with a long lifetime, generally equal to the lifetime of the Expires header in the request (this lifetime could be modified by a different expiration in the response).

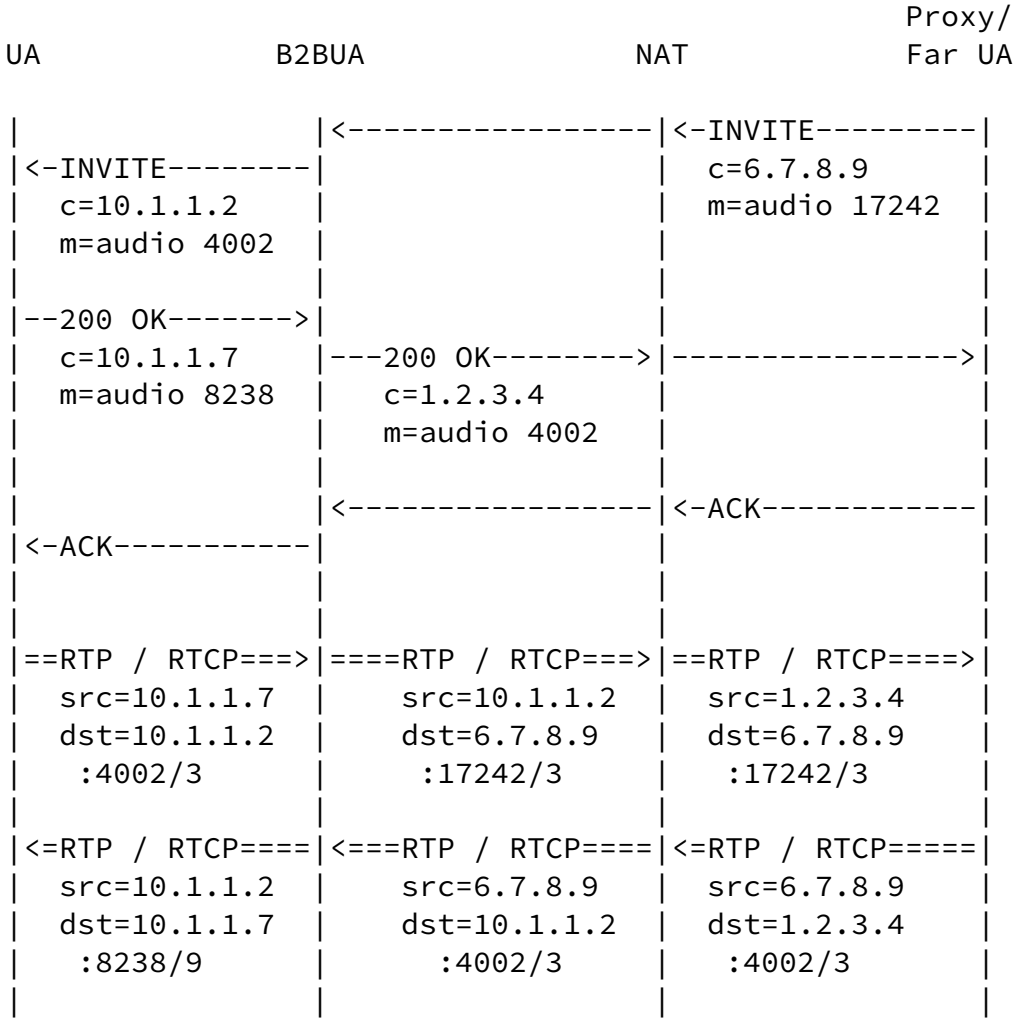
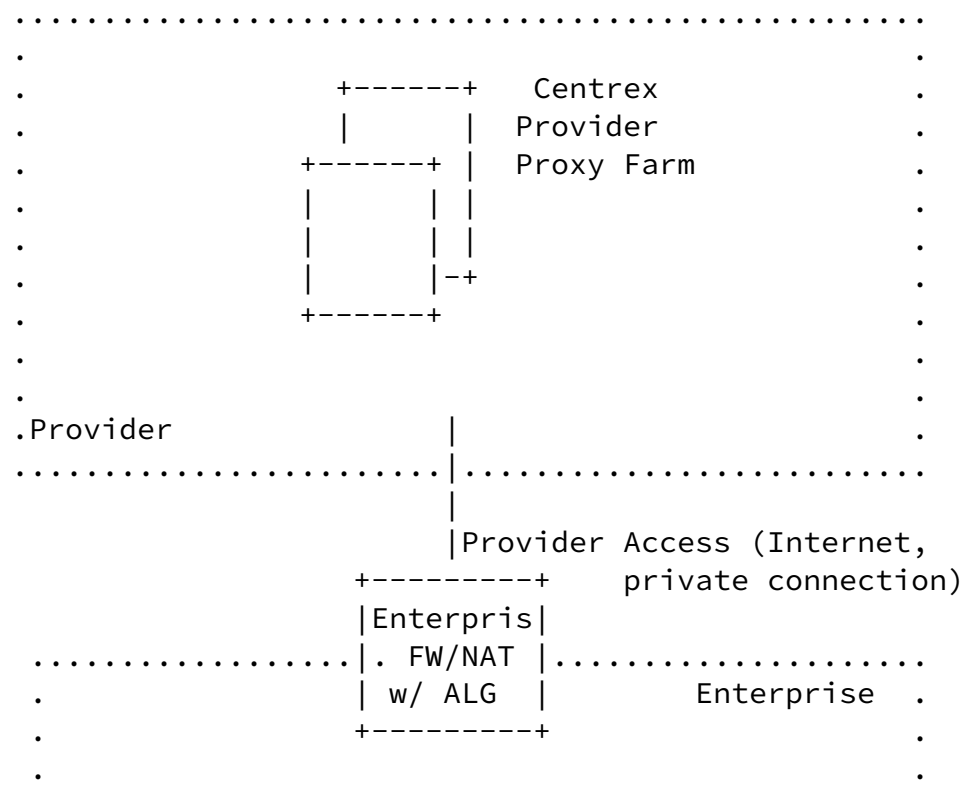


Figure 19: Enterprise with B2BUAWM: Incoming Invite

So, for example, the REGISTER sent by a client (message 1) might look like:

```
REGISTER sip:enterprise.com SIP/2.0
From: sip:user12@enterprise.com
To: sip:user12@enterprise.com
Contact: sip:user12@10.0.1.1:5060
CSeq: 86590 REGISTER
Via: SIP/2.0/UDP 10.0.1.1
Call-ID: 9adjasd88
```

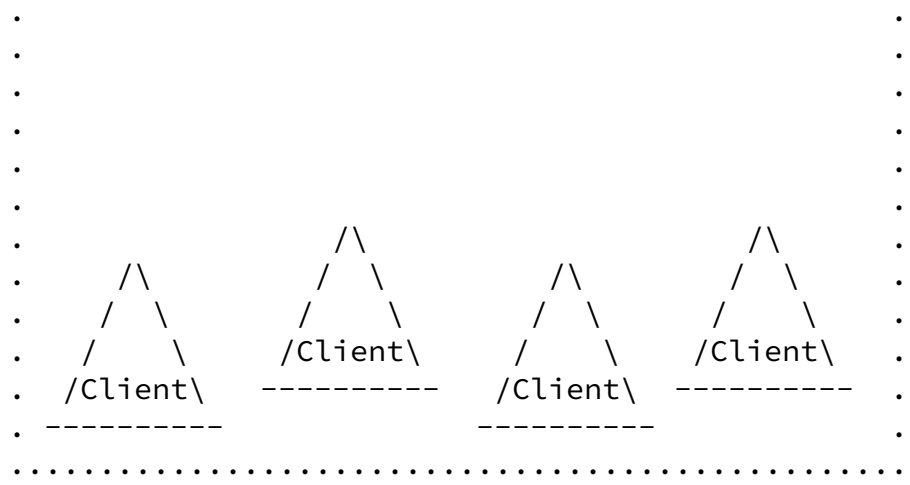
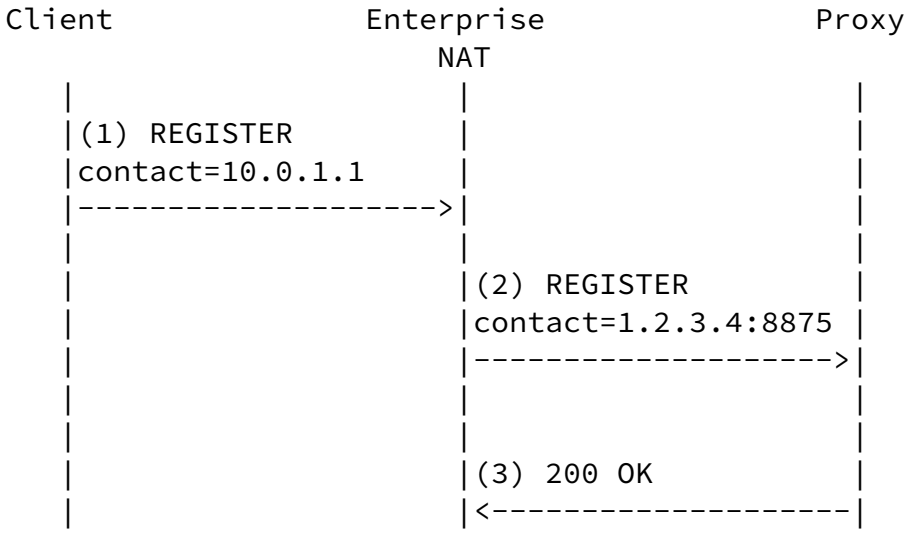


Figure 20: Centrex ALG Configuration

The ALG would rewrite this, and the resulting registration (message 2) might look like:

```
REGISTER sip:enterprise.com SIP/2.0
From: sip:user12@enterprise.com
```

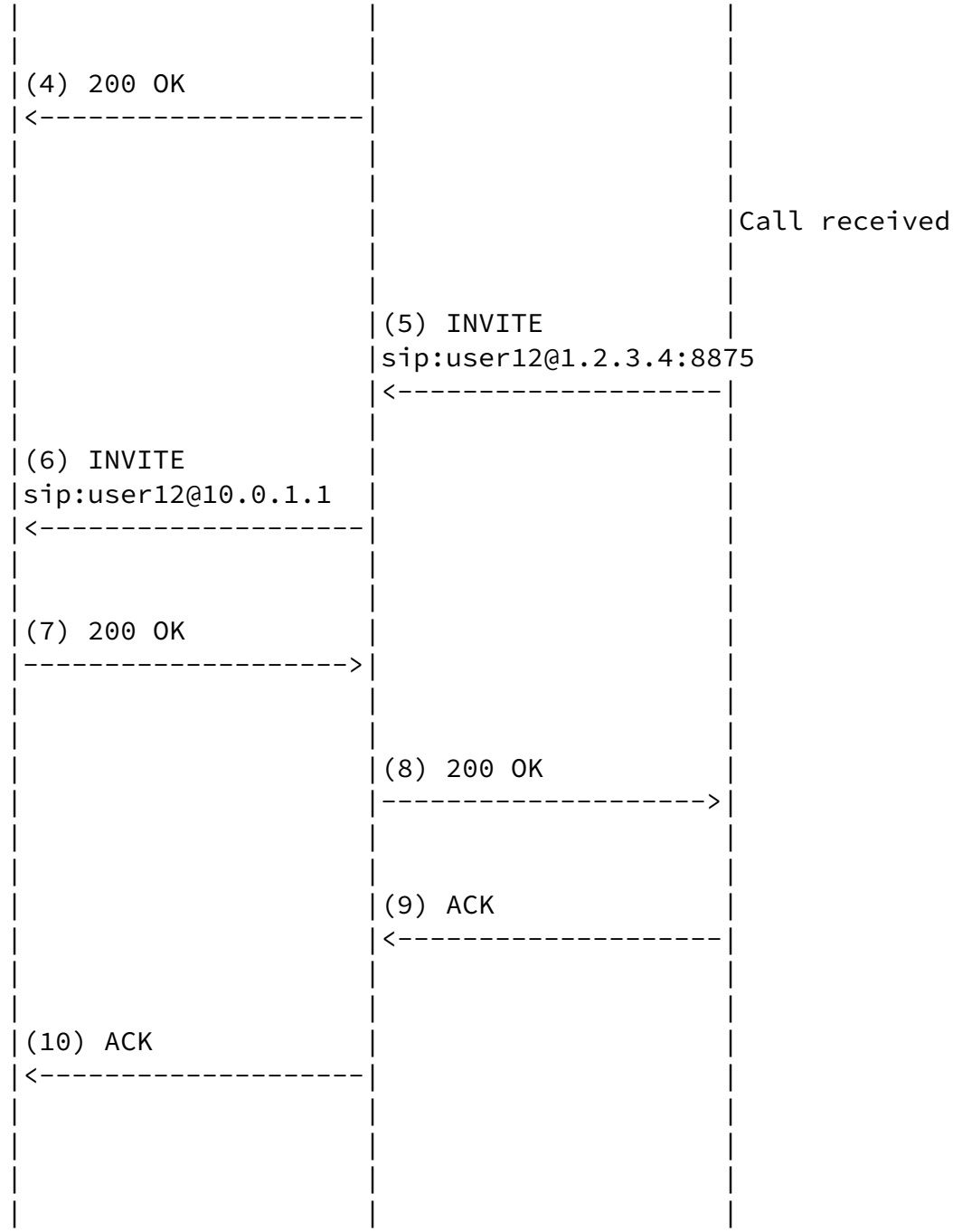


Figure 21: Incoming Call, Centrex ALG

To: sip:user12@enterprise.com
Contact: sip:user12@1.2.3.4:8875
CSeq: 86590 REGISTER

```
Via: SIP/2.0/UDP 1.2.3.4:8875
Call-ID: 9adjasd88
```

The ALG would remember the association (1.2.3.4:8875, 10.0.1.1:5060) for one hour (the default registration expiration). This REGISTER would arrive at the provider proxy, and the registration stored there.

When an INVITE arrives at the provider proxy for sip:user12@enterprise.com, the proxy translates this to sip:user12@1.2.3.4:8875 (message 5), and forwards the INVITE. This arrives at the ALG. The ALG rewrites the request URI to the translated address, sip:user12@10.0.1.1, and forwards it to the client (message 6). The ALG will also need to rewrite the SDP and Contact in the 2xx (message 7).

5.2 Solution II: External B2BUAWM

In this solution, the service provider uses a "Back to Back User Agent with Media" instead of a proxy. This can be viewed as an integration of the proxy and the TURN server into an application specific solution.

With this solution, there is no need for the enterprise to deploy an ALG.

There are two main components of the solution - one addressing the control/signaling path issues, and the other, the media path. The signaling path component consists of a stateful Signaling Proxy server exhibiting some distinct behavior to be discussed later. The media path component consists of a Media Proxy server. In the context of SIP and RTP, the signaling path component will be a SIP Proxy server called Back-to-Back-User-Agent (BBUA) and the media path component is an RTP Proxy, as shown in Figure 22. The signaling and media proxies interact using some control protocol, which is transparent to the end users.

5.2.1 Signaling Path

The following prerequisites must be met in order for the solution to work:

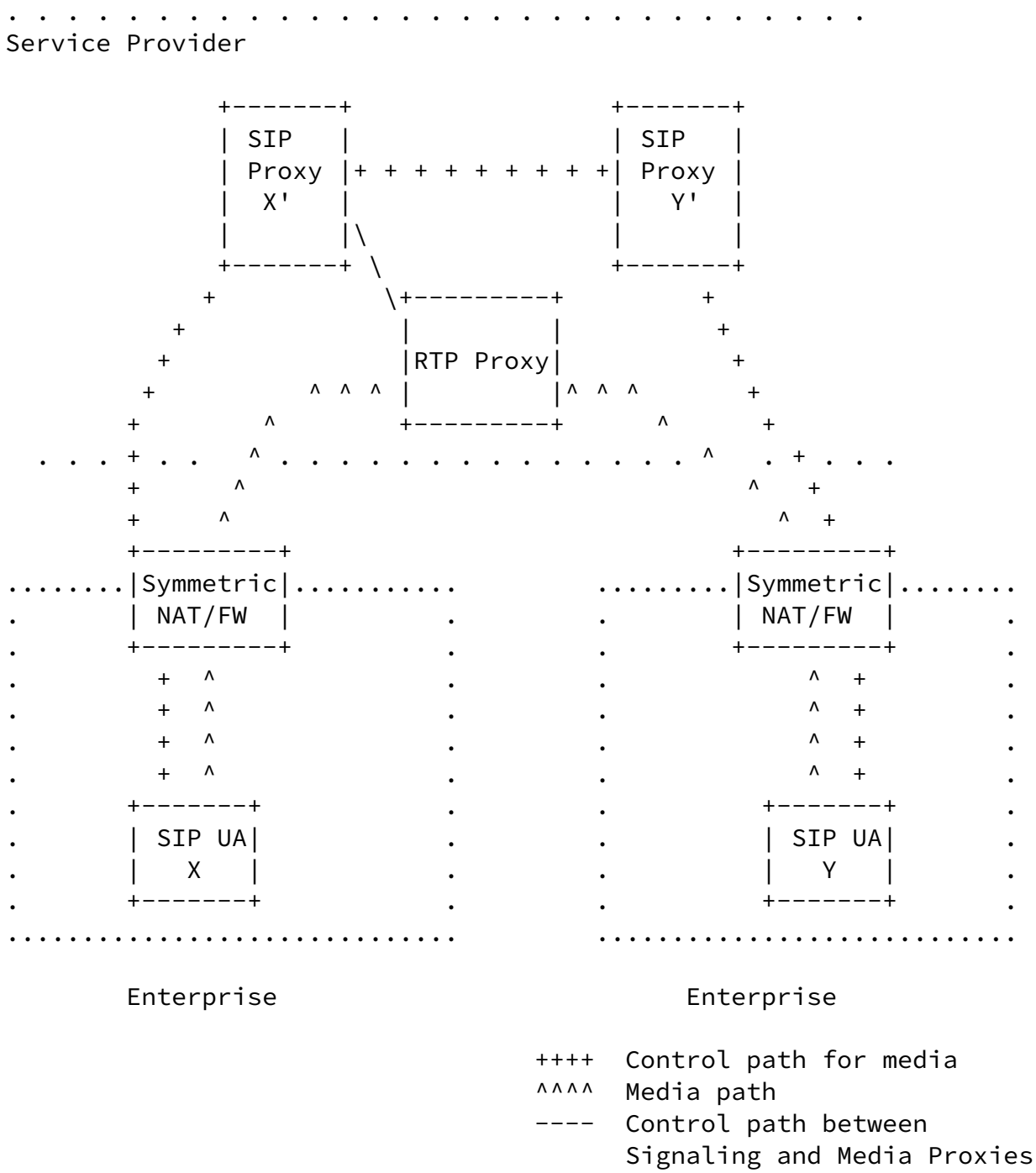


Figure 22: B2BUAWM Configuration

o The solution described here, relies on the principle that connections from within the Enterprise are allowed out and

that the corresponding responses to those connections are allowed back in (as in case of Traditional NAT's). This

enables the end points inside the enterprise to maintain a connection to the Signaling server via a "keep-alive" mechanism.

- o All signaling messages must travel via the Signaling Proxy. The SIP Signaling Proxy server with whom the UA registered must be the one through which all incoming (outgoing) calls are routed to (from) the UA.

- o The SIP Signaling Proxy server must be one hop away from the UA.

- o The destination port of the requests at the Signaling Proxy server is the same as the source port of the corresponding responses.

In the example configuration shown in Figure 22, the signaling path is established when the SIP UA's register with the corresponding Proxy/Register (Signaling server). The SIP REGISTER message contains an extension tag carried in the Proxy-Require header, indicating to the Proxy that the client is behind a NAT (Note: the client or the Proxy does not care about the type of NAT, as the same solution applies to all types). This packet is NAT-ed by the Enterprise NAT with a new source IP address and port. If the Proxy supports the extension tag, it creates an association between the IP address and port from which the packet arrived and the actual address of the user. The response to the Registration message is sent to this IP address and port (not to the Contact address in the REGISTER).

Once the signaling path is established, this can be used to send subsequent requests and responses to the user using the above association stored at the Proxy, i.e., the request is sent (as above) to the address/port assigned by the NAT instead of to the actual address of the user. All requests and responses must be routed via this Signaling Proxy. All requests from the user should carry the Proxy-require header with the special tag indicating that it is behind NAT/FW.

The signaling path is always kept open through the NAT using a keep-

alive mechanism. This can be done using REGISTER refreshes as proposed in [9]. One disadvantage of using the REGISTER method for this purpose is it forces frequent (at least once every minute), unnecessary involvement on the part of the Registrar. In this solution, "keep-alive" is achieved by periodically sending a new lightweight SIP method from the UA to the Server designed specifically for this purpose. The new SIP method is called PING and contains the following mandatory SIP headers: From, To, Via, Proxy-require, Contact, Call-id, Cseq. The Proxy Server responds to the

PING method with a 200 OK final response. The timeout values in Firewalls and NAT's are observed to be between 1 to 3 mins and the "keep-alive" frequency must be higher than this. This keep-alive mechanism must continue during the call.

The public IP address/port allocated by the NAT is returned to the Client in the Contact header of 200 OK sent by the Proxy server. This allows the UA to perform NAT/FW failure detection by - (a) not receiving 200 OK over a prolonged period of time, and (b) detecting that the NAT IP address has changed in a received 200 OK. The second event can serve as an indication that the UA needs to re-register with the Proxy server to maintain the signaling path open. The NAT address/port, although sent to the registered UA, is never carried in SIP messages towards the remote callee. This provides certain amount of security through IP address hiding.

5.2.2 Media Path: The Case for RTP (Media) Proxy

RTP Proxy, as the name suggests, terminates an RTP session on one side and originates the same session on the other. The UA always sends (and receives) media to (and from) an assigned address and port of the RTP proxy. The RTP Proxy can perform NAPT function both on the source and destination address/port of packets. For a solution using an RTP Proxy to work, the requirement is that any media stream traversing the NAT from the private side (e.g., Enterprise) must always go through the RTP Proxy (and any intra-Enterprise traffic should be able to bypass the RTP Proxy). Since incoming packets are received from the same address and port as the outgoing packets are sent to, a Symmetric NAT will always allow the packets from the RTP proxy inside the Enterprise for the duration of the session. This implies that a Full Cone NAT (or any restricted version of it) will also allow flows to and from the RTP Proxy. This principle can be

generalized to be applicable to any type of application session, not just RTP/RTCP sessions.

The following are the prerequisites for this solution to work:

- Any media stream that traverses the NAT from private side to the public will pass through the RTP Proxy.
- The addresses and ports in the RTP Proxy are assigned either during the call set-up or before it.
- The RTP Proxy should know where to send the received media. For example, if it is acting as a bridge between two private endpoints X and Y (X and Y are behind different NAT's), this implies that the Proxy should be aware of the X's external address/port before (or at least, as soon as) it starts

receiving media from Y, and vice-versa.

- The send and receive ports of the media in the UA's are configured to be the same.

Reverting back to our example configuration in Figure 22, let us illustrate using a call flow, how resource allocation in the RTP Proxy will take place when UA X wants to establish a SIP session with UA Y (only relevant parts of the call flow is shown). It is assumed that the signaling paths between the UA's and the Proxy are already established by the method described in Section 5.2.1. Please refer to Figure 23 for the address/port allocations at various devices and the call flow.

1. UA X sends a SIP INVITE message towards UA Y through the Proxy X' specifying that it wishes to receive media at (private) IP address, PXA and port, px, i.e., PXA:px.
2. Proxy X' instructs the RTP Proxy to reserve a pair of IP address/port, one representing each end of the connection. To UA Y, the remote end of the connection is perceived to be A:px*, whereas to UA X, this is perceived to be A:py*. Note 1: Due to the presence of the NAT's, the source IP address/port of the media packets from the UA's are yet

unknown to the RTP Proxy. Note 2: Consecutive port binds are also created for RTCP sessions corresponding to RTP.

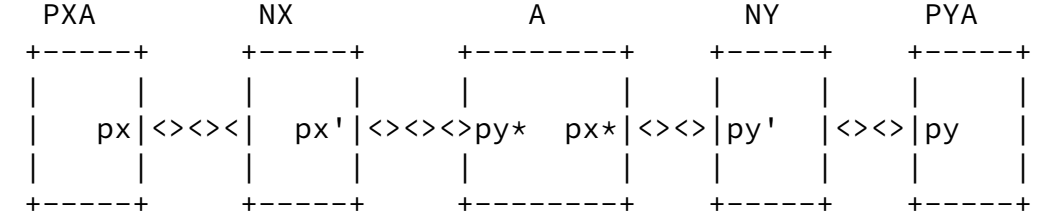
3. Proxy X' forwards the INVITE towards UA Y (perhaps through Proxy Y') specifying that Y should send media to the RTP Proxy at A:px* (by modifying the SDP).

4. UA Y responds with a 200 OK specifying that it wishes to receive media at (private) IP address, PYA and port, py, i.e., PYA:py.

5. When Proxy X' receives the 200 OK, it changes these IP address and port parameters in SDP specifying X should transmit media to the RTP Proxy at A:py*, and forwards the 200 OK towards UA X (Note: Proxy X' actually forwards the message to the external NAT-ed address/port of UA X).

6. When UA X receives the 200 OK, it starts transmitting media (e.g., background noise) towards the RTP Proxy. When the first of these NAT- ed RTP packets reach the RTP Proxy, the Proxy remembers the source address/port (say, NX, px') of that packet as the external representation for the media end point of UA X. Any media received from Y to X should be

Legends:-
<><> : bi-directional flow
---- : signaling
==== : media
px, px', py*, px*, py', py : ports
PXA, NX, A, NY, PYA : IP addresses



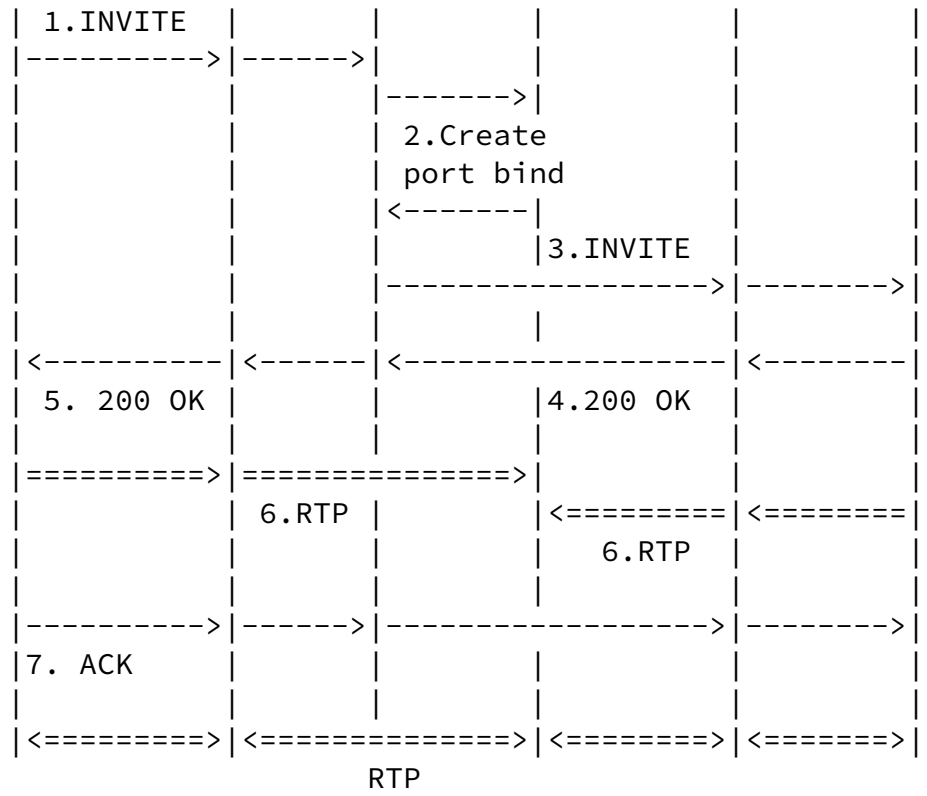


Figure 23: Call Flow for B2BUAWM

sent to this address/port.

Similarly, when the first RTP packet is received from UA Y,

the RTP Proxy notes down the source IP address/port (say, NY:py'), which will be used as the destination address to transmit media received from X to Y.

7. Finally, UA X responds with an ACK message and the call set-up is complete.

Note that, since all signaling is routed via the Proxy, which can determine whether both the UA's are in the same domain, all intra-Enterprise calls (behind the same NAT) can avoid the trip to the RTP

Proxy. This is one of the key advantages of the Proxy mediating the address/port of the endpoints of a call.

Once the media path is established through the NAT, keep-alive messages in the form of periodic RTP packets are sent to keep the connection alive when the users are in mute (i.e., when no speech packets are transmitted).

There needs to be some control interaction between the SIP and the RTP Proxies to establish the end-to-end sessions. The protocol can be one of the device control protocols such as MGCP, Megaco etc.

5.3 Solution III: Outsourced MIDCOM

In this scenario, the enterprise upgrades its firewall/NAT to a midcom-enabled one. The centrex provider uses a proxy which can control the firewall/NAT, opening and closing pinholes as needed. This scenario avoids the need for a SIP specific ALG. Indeed, if the enterprise uses other outsourced services, each provider can have a control connection to the firewall/NAT to ensure that it works for that particular application. This is one of the benefits of MIDCOM.

This configuration (for a single centrex provider) is shown in Figure 24. It is similar to the ALG configuration 20, but there is a control relationship between the proxies in the provider network (which act as midcom agents [7]), and the firewall/NAT in the enterprise (which is the middlebox).

For this solution to work, the firewall/NAT needs a static rule which allows all outgoing traffic on UDP port 5060 to the proxies in the provider network. This rule can be configured, or the proxies in the provider network can create it when they first connect to the firewall/NAT.

The basic call flow for a registration and incoming INVITE is shown in Figure 25. The enterprise is using a MIDCOM FW/NAT. The enterprise

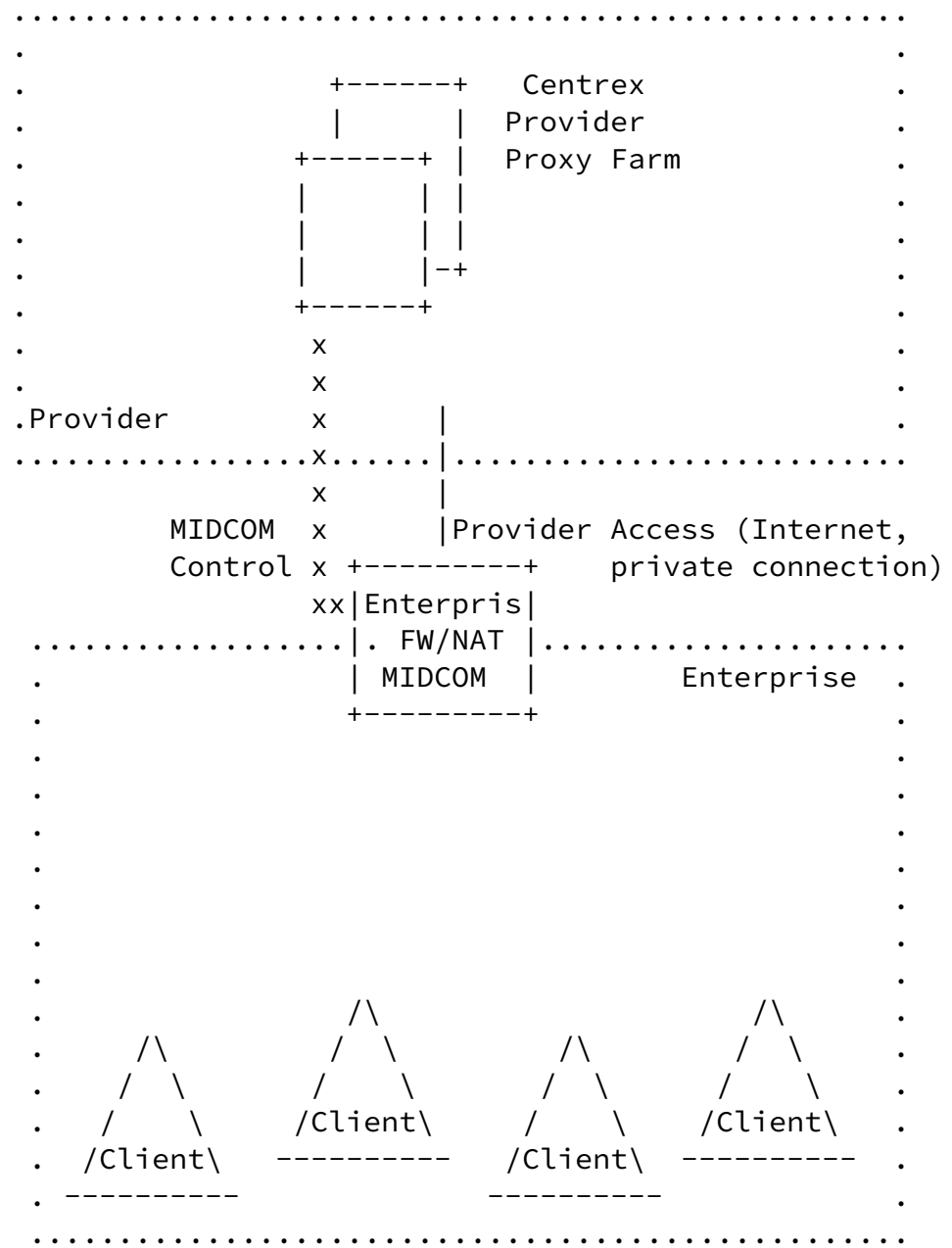
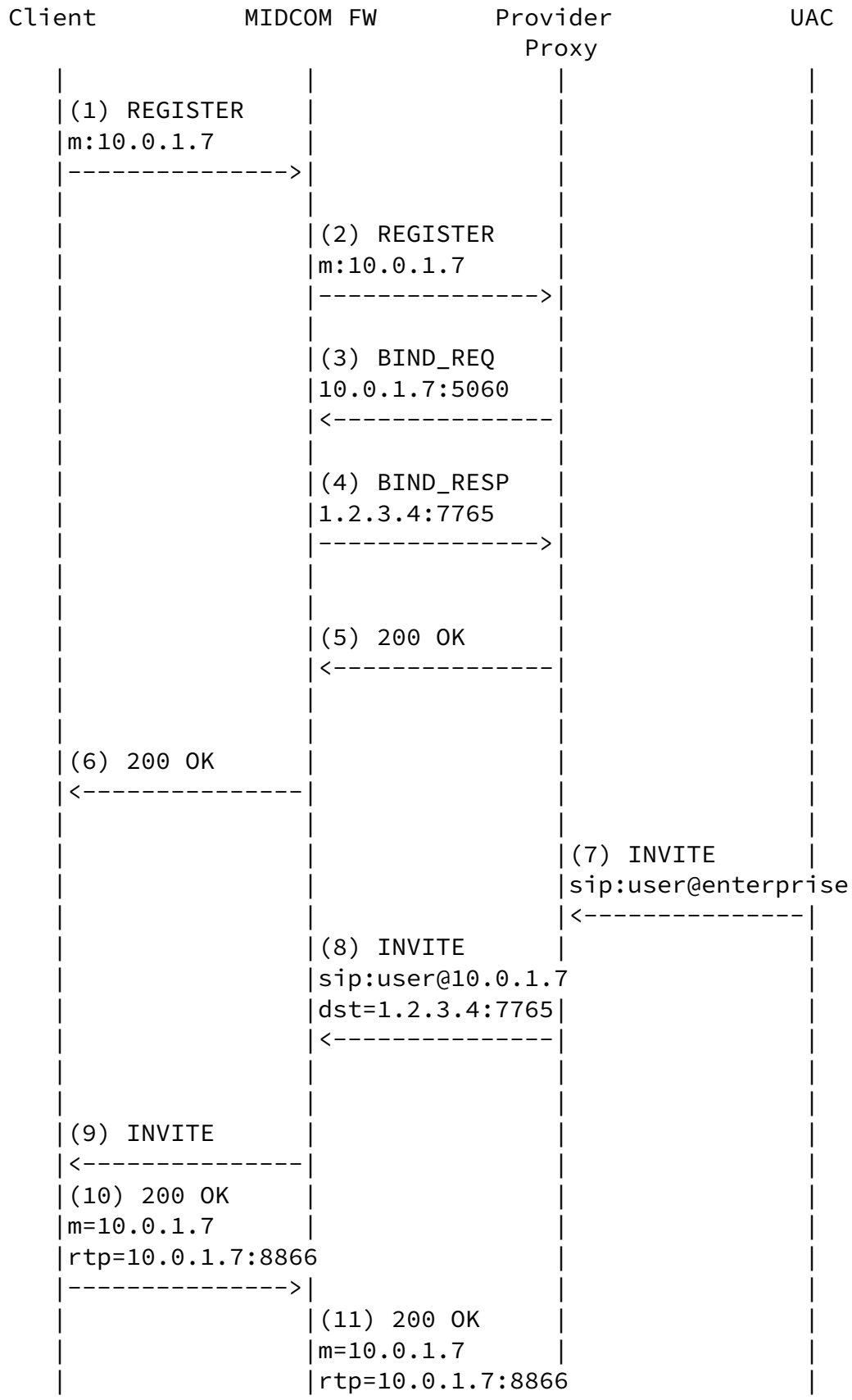


Figure 24: Centrex MIDCOM Configuration

has a single public address, 1.2.3.4, which routes to the FW/NAT. When the REGISTER arrives at the proxy (message 2), the proxy performs a bind request (message 3), asking for the Contact address, 10.0.1.7:5060, to be translated. The FW/NAT returns with a mapped address, 1.2.3.4:7765, and this is stored by the proxy to use as the destination IP/port for sending requests to sip:user@10.0.1.7.


Client
MIDCOM FW
Provider Proxy
UAC
(1) REGISTER m:10.0.1.7
(2) REGISTER m:10.0.1.7
(3) BIND_REQ 10.0.1.7:5060
(4) BIND_RESP 1.2.3.4:7765
(5) 200 OK
(6) 200 OK
(7) INVITE sip:user@enterprise
(8) INVITE sip:user@10.0.1.7 dst=1.2.3.4:7765
(9) INVITE
(10) 200 OK m=10.0.1.7 rtp=10.0.1.7:8866
(11) 200 OK m=10.0.1.7 rtp=10.0.1.7:8866

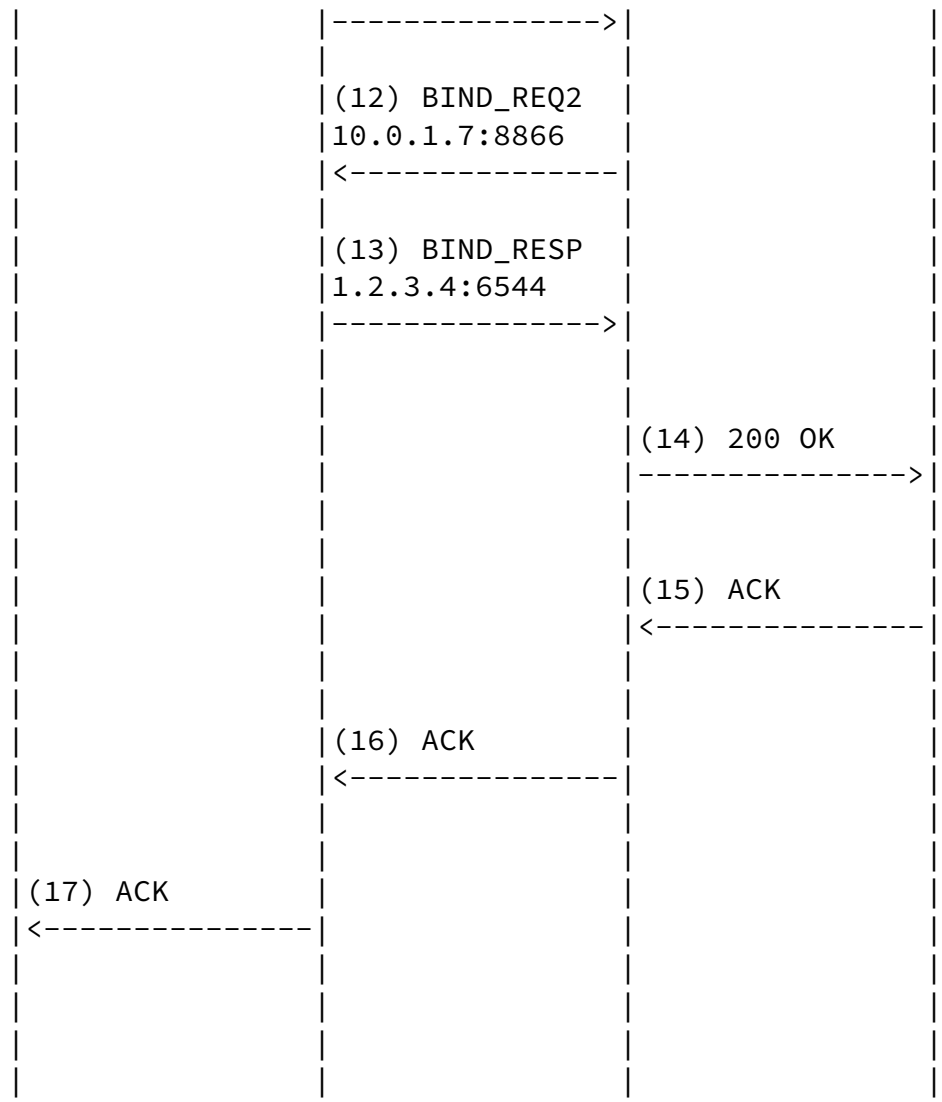


Figure 25: Centrex MIDCOM Solution, Incoming INVITE

When an incoming INVITE arrives at the proxy (message 7), the R-URI is looked up, and translated to sip:10.0.1.7, but with a destination address of 1.2.3.4:7765, which is where the request is sent. This passes through the NAT, translated to 10.0.1.7:5060, and arrives at the client (message 9). The 200 OK contains an RTP address of 10.0.1.7:8866. When this is received by the proxy (11), it performs another bind request to the FW/NAT, asking for a pair of addresses on consecutive ports. The bind response indicates that 1.2.3.4:6544 and 1.2.3.4:6545 will be mapped to 10.0.1.7:8866 and 10.0.1.7:8867 respectively. The proxy rewrites the SDP in the response, and

forwards that upstream.

5.4 Solution IV: STUN and TURN

The operation of a STUN or TURN solution in the centrex case is identical to the residential case described in Section 2.2. If the enterprise uses a firewall/NAT which allows for full-cone operation of UDP, then STUN will get used, and the centrex provider needs to only deploy STUN servers. If, however, a more restrictive firewall/NAT, such as symmetric, is used, the centrex provider must deploy TURN servers as well.

6 Future Work

- Define a set of criteria for evaluating the various solutions.
- Include a subsection at the end of each scenario, weighing the pros/cons of each solution based on the criteria above.
- Expand the scope of the solution set to include a broader set of solutions.
- More details on many of the call flows.

7 Acknowledgements

The authors would like to thank Mary Barnes for her comments on this draft.

8 Authors Addresses

Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936

email: jdrosen@dynamicsoft.com

Rohan Mahy

Cisco Systems
170 West Tasman Dr, MS: SJC-21/3
Phone: +1 408 526 8570
Email: rohan@cisco.com

Sanjoy Sen
Nortel Networks
sanjoy@nortelnetworks.com

9 Bibliography

[1] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: session initiation protocol," Request for Comments 2543, Internet Engineering Task Force, Mar. 1999.

[2] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, "STUN - simple traversal of UDP through NATs," Internet Draft, Internet Engineering Task Force, Oct. 2001. Work in progress.

[3] J. Rosenberg, R. Mahy, and C. Huitema, "Traversal using nat relay (turn)," Internet Draft, Internet Engineering Task Force, Nov. 2001. Work in progress.

[4] B. Biggs, "A SIP application level gateway for network address translation," Internet Draft, Internet Engineering Task Force, Mar. 2000. Work in progress.

[5] C. Martin and A. Johnston, "SIP through NAT enabled firewall call flows," Internet Draft, Internet Engineering Task Force, Feb. 2001. Work in progress.

[6] J. Rosenberg, D. Drew, and H. Schulzrinne, "Getting SIP through firewalls and NATs," Internet Draft, Internet Engineering Task Force, Feb. 2000. Work in progress.

[7] P. Srisuresh, J. Kuthan, J. Rosenberg, A. Molitor, and A. Rayhan, "Middlebox communication architecture and framework," Internet Draft, Internet Engineering Task Force, Oct. 2001. Work in progress.

[8] C. Huitema, "RTCP attribute in SDP," Internet Draft, Internet Engineering Task Force, Aug. 2001. Work in progress.

[9] J. Rosenberg, J. Weinberger, and H. Schulzrinne, "SIP extensions for NAT traversal," Internet Draft, Internet Engineering Task Force, Aug. 2001. Work in progress.

[10] M. Borella, D. Grabelsky, J. Lo, and K. Taniguchi, "Realm specific IP: protocol specification," Request for Comments 3103, Internet Engineering Task Force, Oct. 2001.

[11] M. Borella, J. Lo, D. Grabelsky, and G. Montenegro, "Realm specific IP: framework," Request for Comments 3102, Internet Engineering Task Force, Oct. 2001.

[12] F. Thernelius, "SIP firewall solution," Internet Draft, Internet Engineering Task Force, July 2000. Work in progress.

[13] S. Kent and R. Atkinson, "IP encapsulating security payload (ESP)," Request for Comments 2406, Internet Engineering Task Force, Nov. 1998.

[14] M. Gaynor and S. Bradner, "Firewall enhancement protocol (FEP)," Request for Comments 3093, Internet Engineering Task Force, Apr. 2001.

[15] D. Yon, "Connection-oriented media transport in SDP," Internet Draft, Internet Engineering Task Force, Oct. 2001. Work in progress.

[16] M. Arango, A. Dugan, I. Elliott, C. Huitema, and S. Pickett, "Media gateway control protocol (MGCP) version 1.0," Request for Comments 2705, Internet Engineering Task Force, Oct. 1999.

[17] F. Cuervo, N. Greene, A. Rayhan, C. Huitema, B. Rosen, and J. Segers, "Megaco protocol version 1.0," Request for Comments 3015, Internet Engineering Task Force, Nov. 2000.

Table of Contents