SIPPING
Internet-Draft
Expires: January 17, 2005

J. Rosenberg
dynamicsoft
G. Camarillo
Ericsson
July 19, 2004

Examples of Network Address Translation (NAT) and Firewall Traversal for the Session Initiation Protocol (SIP) draft-rosenberg-sipping-nat-scenarios-03

Status of this Memo

Copyright Notice

Abstract

This document contains a set of examples about how to establish sessions through Network Address Translators (NATs) using the Session Initiation Protocol (SIP). NAT traversal for SIP is accomplished using Interactive Connectivity Establishment (ICE), which allows the media streams to work, in addition to the SIP extension for symmetric response routing, which allows SIP itself to flow through NAT. The examples cover a range of network topologies and use cases. This

variability helps to demonstrate that the ICE methodology always works, and that a common client algorithm, independent of the network topology and deployment configuration, results in the best connectivity.

Table of Contents

1. Introduction

The Session Initiation Protocol (SIP) [1], without any extensions, has difficulty in networks that contain Network Address Translators (NAT). SIP, out of necessity, breaks many of the guidelines described in RFC 3235 [2]. NAT traversal for SIP is especially problematic for the media streams, which generally flow from user agent to user agent.

To remedy this, RFC 3581 [3] defines a SIP extension for symmetric response routing, which allows SIP itself to traverse NAT. In order for the media streams to traverse NAT, Interactive Connectivity Establishment (ICE) [4] is used. ICE describes a methodology for NAT traversal for multimedia signaling protocols, such as SIP. It also defines some extensions to the Session Description Protocol (SDP) [5] for conveying additional data. ICE makes use of several protocols, namely the Simple Traversal of UDP Through NAT (STUN) [6] and Traversal Using Relay NAT [7], in order to operate.

This document contains a number of example deployment topologies and network configurations. For each, it shows how clients compliant to the above specifications will properly establish communications, and indeed, will do so using the optimal media path for that scenario. This document focuses on media streams that are carried over the Real Time Transport Protocol (RTP) [8]. In all cases, only RTP is shown and discussed, to simplify the discussion. RTCP related operations (generally STUN queries parallel to the RTP ones) are omitted.

2. Residential Users

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.

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.

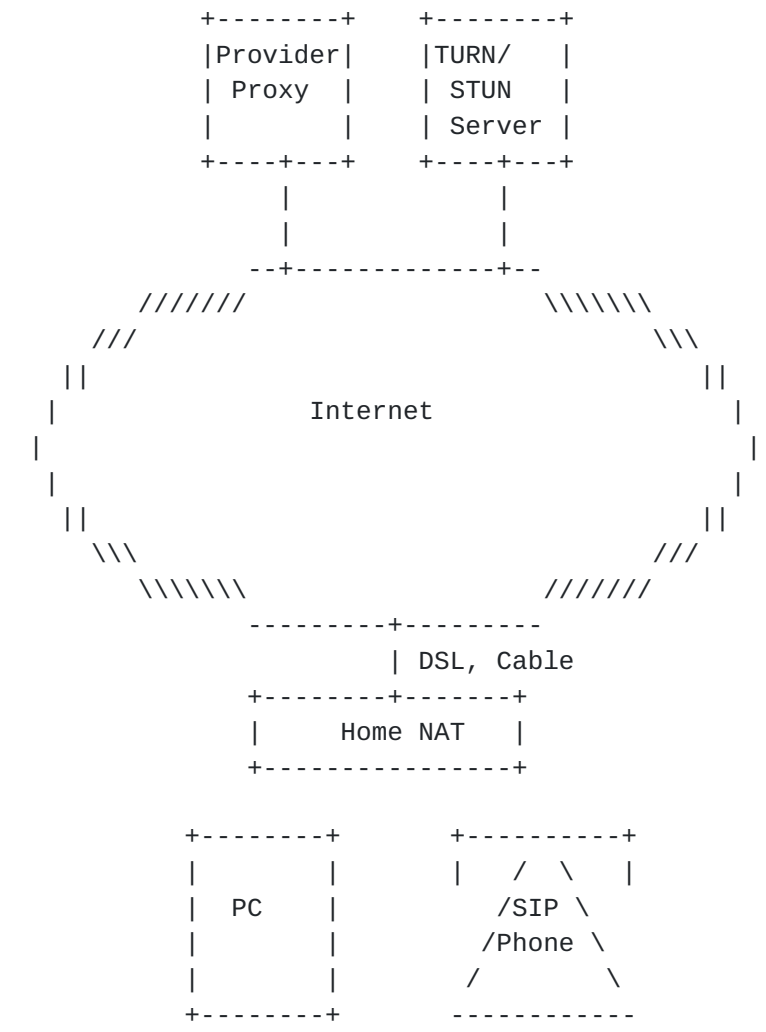
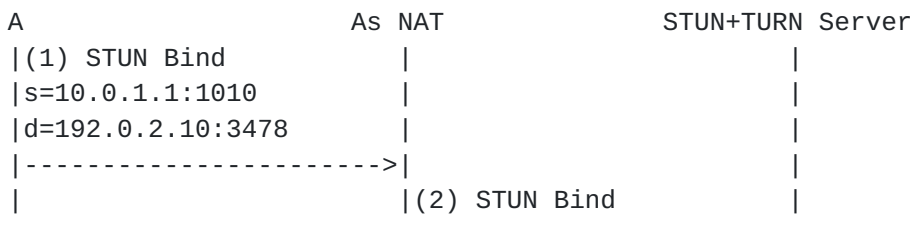


Figure 1: Residence with Single NAT

In this case, the provider administers a SIP proxy and a TURN/STUN server. This server is running STUN on the default port (3478) and TURN on port 5556.

2.1 Full Cone NAT



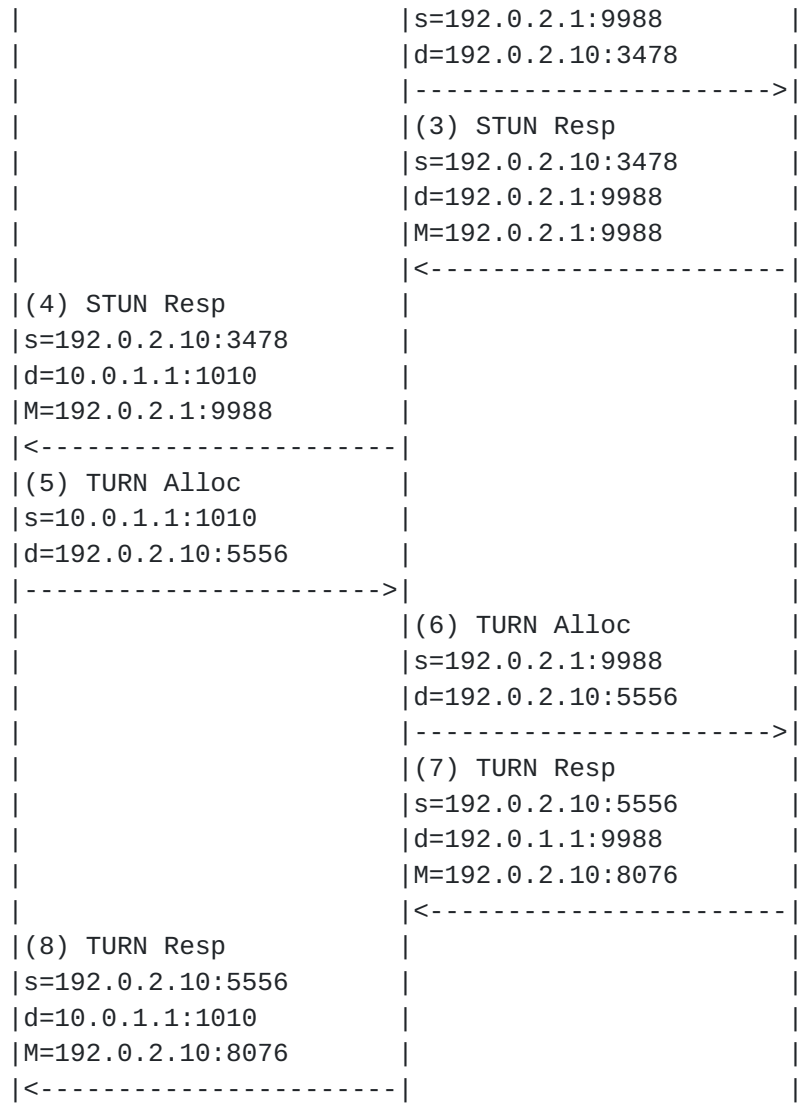


Figure 2: Message sequence for A's Unilateral Allocations

We first consider the case where two such residential users call each other, and both are using NATs of the full-cone variety. The caller follows the ICE algorithm. As such, it firsts allocates a pair of ports on its local interface for RTP and RTCP traffic (10.0.1.1:1010 and 10.0.1.1:1011). As shown in Figure 2, the client issues a STUN request from the RTP port (message 1), which passes through the NAT on its way to the STUN server. In the figure, the "s=" indicates the source transport address of the message, and "d=" indicates the destination transport address. The NAT translates the 10.0.1.1:1010 to 192.0.2.1:9988, and this request arrives at the STUN server (message 2). The STUN server copies the source address into the MAPPED-ADDRESS field in the STUN response (the M= line in message 3), and this passes through the NAT, back to the client. The client now

has a STUN derived transport address of 192.2.0.1:9988. Thought not show, the client will follow a similar process to obtain a STUN derived transport address for RTCP. However, this address will frequently not occupy an adjacent port to the RTP.

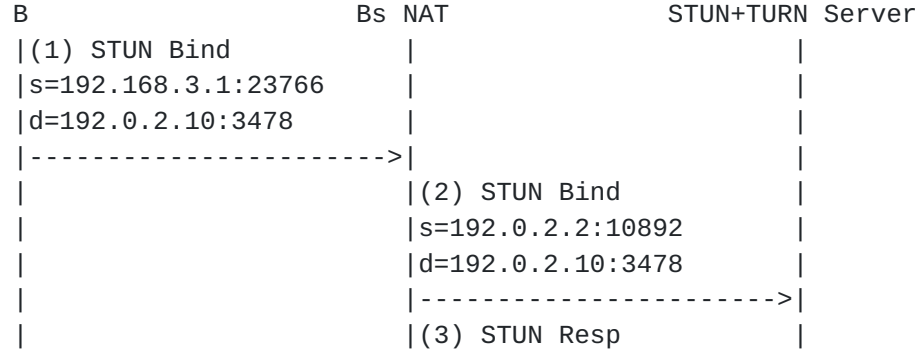
Next, the client follows a similar process to obtain a TURN port for RTP (messages 5-8). The TURN requests are also sent from the same local transport address. Note, however, that the TURN derived transport addresses for RTP (192.0.2.10:8076) and RTCP will be on adjacent ports. This is because the TURN pre-allocation procedure was used in the TURN request for the RTP port (message 5).

The client prioritizes these addresses, choosing the local interface address with priority 1.0, the STUN address with priority 0.8, and the TURN address with priority 0.4. From this, it generates an offer that looks like this:

```
v=0
o=alice 2890844730 2890844731 IN IP4 host.example.com
s=
c=IN IP4 192.0.2.10
t=0 0
m=audio 8076 RTP/AVP 0
a=alt:1 1.0 : user 9kksj== 10.0.1.1 1010
a=alt:2 0.8 : user1 9kksk== 192.0.2.1 9988 192.0.2.1 9990
a=alt:3 0.4 : user2 9kksl== 192.0.2.10 8076
```

Figure 3: A's Offer

Note how the TURN derived transport address is used in the m and c lines, since this is the address with the highest probability of working with a non-ICE peer. That address is also included in the list of alteratives (with ID 3). Also note that because the STUN derived transport address for RTP and RTCP were not adjacent, two transport addresses are provided for alternate 2.



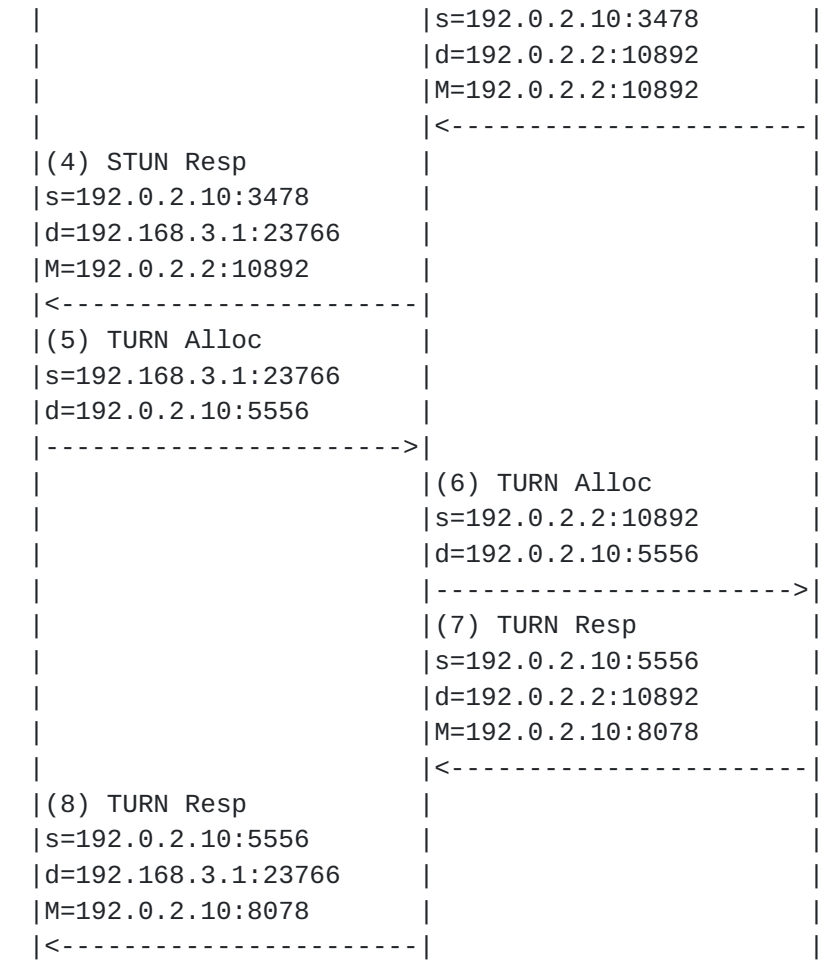
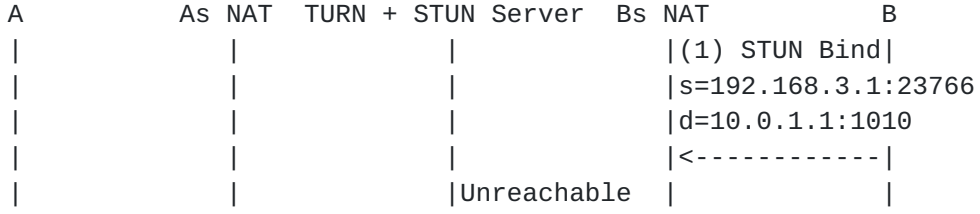
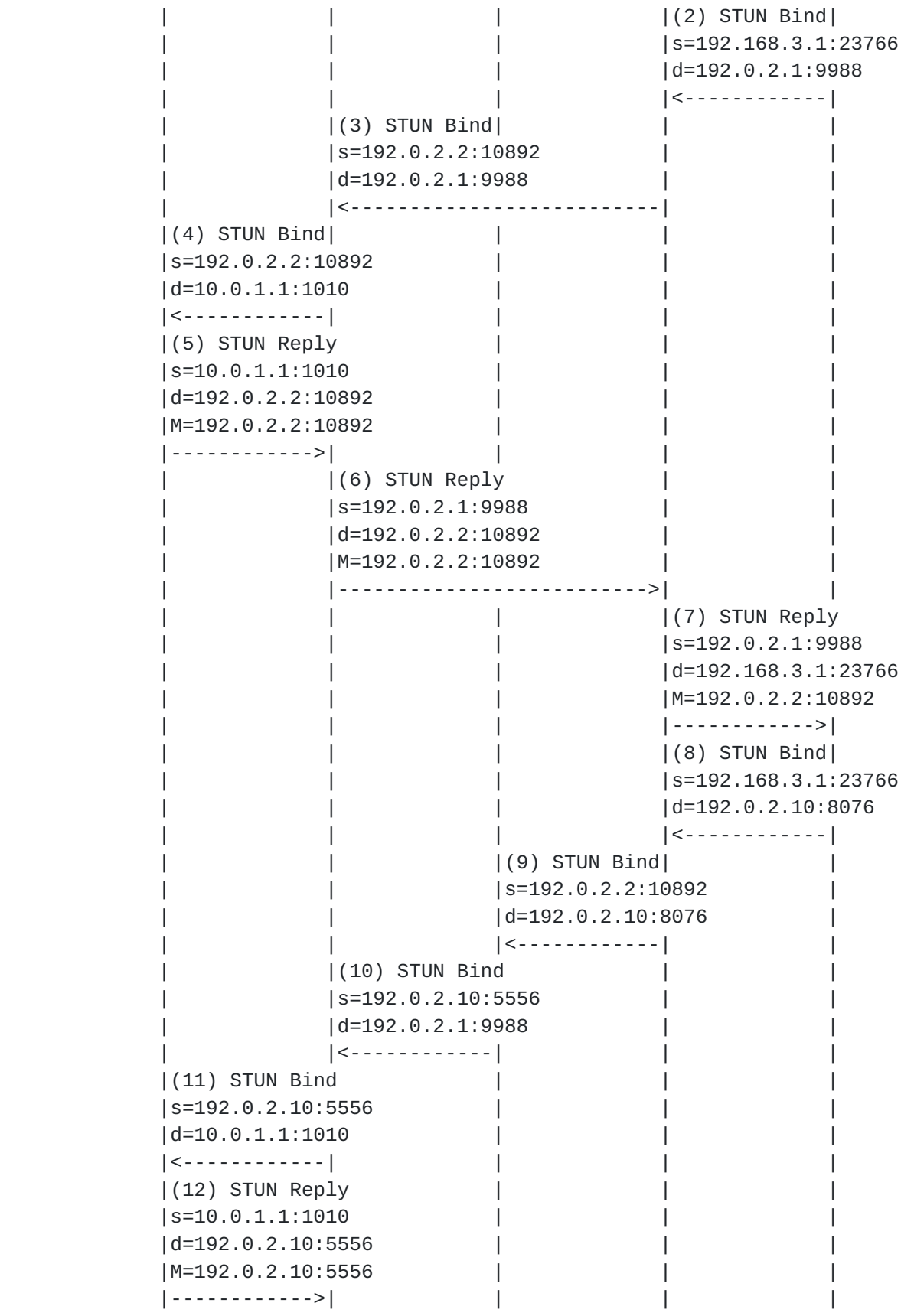


Figure 4: Message sequence for B's Unilateral Allocations

This offer arrives at the called party, user B. User B is also behind a full-cone NAT, and is using the 192.168/16 private address space internally. It happens to be using the same service provider as A, and is therefore using the same TURN server, at 192.0.2.10:5556. User B follows the same set of procedures followed by user A. It uses local interfaces, STUN, and TURN, and obtains a set of transport addresses that it can use. This process is shown in Figure 4. This process differs from that of Figure 2 only in the actual addresses and ports used and obtained.




(2) STUN Bind
s=192.168.3.1:23766
d=192.0.2.1:9988
(3) STUN Bind
s=192.0.2.2:10892
d=192.0.2.1:9988
(4) STUN Bind
s=192.0.2.2:10892
d=10.0.1.1:1010
(5) STUN Reply
s=10.0.1.1:1010
d=192.0.2.2:10892
M=192.0.2.2:10892
(6) STUN Reply
s=192.0.2.1:9988
d=192.0.2.2:10892
M=192.0.2.2:10892
(7) STUN Reply
s=192.0.2.1:9988
d=192.168.3.1:23766
M=192.0.2.2:10892
(8) STUN Bind
s=192.168.3.1:23766
d=192.0.2.10:8076
(9) STUN Bind
s=192.0.2.2:10892
d=192.0.2.10:8076
(10) STUN Bind
s=192.0.2.10:5556
d=192.0.2.1:9988
(11) STUN Bind
s=192.0.2.10:5556
d=10.0.1.1:1010
(12) STUN Reply
s=10.0.1.1:1010
d=192.0.2.10:5556
M=192.0.2.10:5556

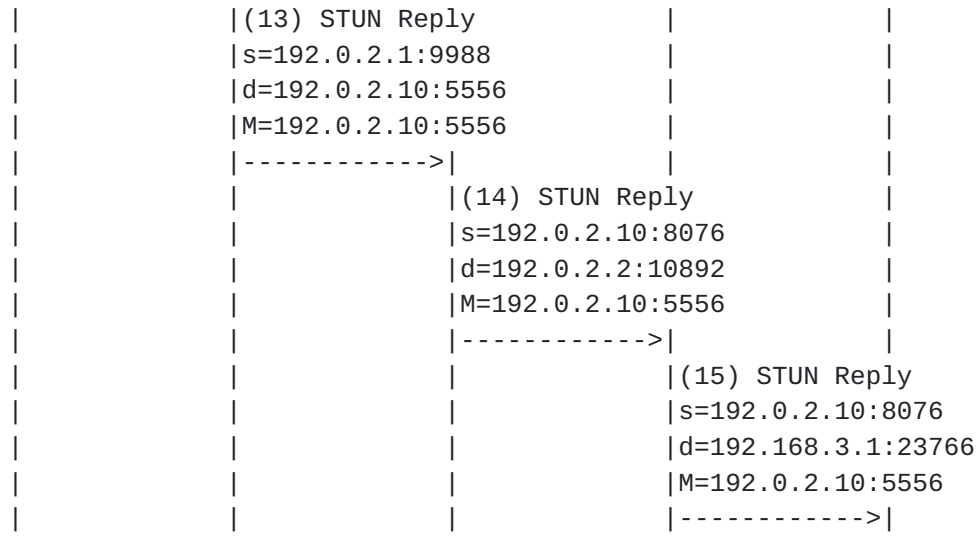


Figure 5: B's Connectivity Checks

While B's phone is ringing, B's user agent uses STUN to test connectivity from its local transport address pair (192.168.3.1:23766 and 192.168.3.1:23767) to the three alternates listed in the offer. The flow for that is shown in Figure 5. This flow, and the discussions, only consider the RTP transport addresses. The procedures would all be identical for RTCP. First, B tests connectivity to the alternate with ID 1, which is 10.0.1.1:1010. It does so by attempting to send a STUN request to this address (message 1). Of course, this is a private address, and not in the same network as B. Therefore, it is unreachable, and no STUN response is received.

In parallel, B tests connectivity to the alternate with ID 2, which is 192.0.2.1:9988. To do this, it sends a STUN request to that address. It sends it with a source address equal to its local transport address; the same one that it used to send the previous TURN and STUN packets (192.168.3.1:23766). This request (message 2) arrives at the NAT. Since the NAT is full cone, and since this address has an existing binding, the NAT translates the source address to that existing binding, 192.0.2.2:10892. This request (message 3) continues onwards to A's NAT. Since A's NAT is also full cone, the existing binding for 192.0.2.1:9988 is used, and the destination address is translated to 10.0.1.1:1010 and then forwarded towards A (message 4). A receives this. It verifies the username and password, and then generates a response. The response contains a MAPPED-ADDRESS equal to the source address seen in the STUN request (192.0.2.2:10892). It passes back through A's NAT (message 5), through B's NAT (message 6), and back to B (message 7).

B examines the MAPPED-ADDRESS in the STUN response. Its

192.0.2.2:10892. However, this is not a new address. B is already aware of this address as a result of its initial STUN Binding requests to the TURN/STUN server (Figure 4). As such, no additional addresses were learned.

In parallel with the tests against ID 2, B tests connectivity to the alternate with ID 3. This is the address A allocated through TURN. Of course, B does not know this. B sends a STUN request to this address (192.0.2.10:8076), and sends it from the same local transport address (192.168.3.1:23766) (message 8). The NAT, once again, translates the source address to 192.0.2.2:10892 (message 9). This is routed to the TURN server. The TURN server locks down the binding allocated to A, such that it will now begin relaying packets sent from A to 192.0.2.2:10892. The TURN server forwards the packet towards A (message 10). This reaches A's NAT, which translates the destination address based on the existing binding. The STUN request is then delivered to A (message 11). A verifies the username and password, and then generates a STUN response. This response contains the source address that the request came from. In this case, that source address is the public transport address of the TURN server (192.0.2.10:5556). This STUN response is relayed all the way back to B (messages 12-15).

B examines the MAPPED-ADDRESS in this STUN response. It's 192.0.2.10:5556, which is a new address. As a result, B has now obtained a peer derived STUN address. It adds this to its list of transport addresses. Its priority equals that of the address it was derived from - ID 3 - which has a qvalue of 0.4.

At some point, B picks up, and an answer is generated. The answer would look like this:

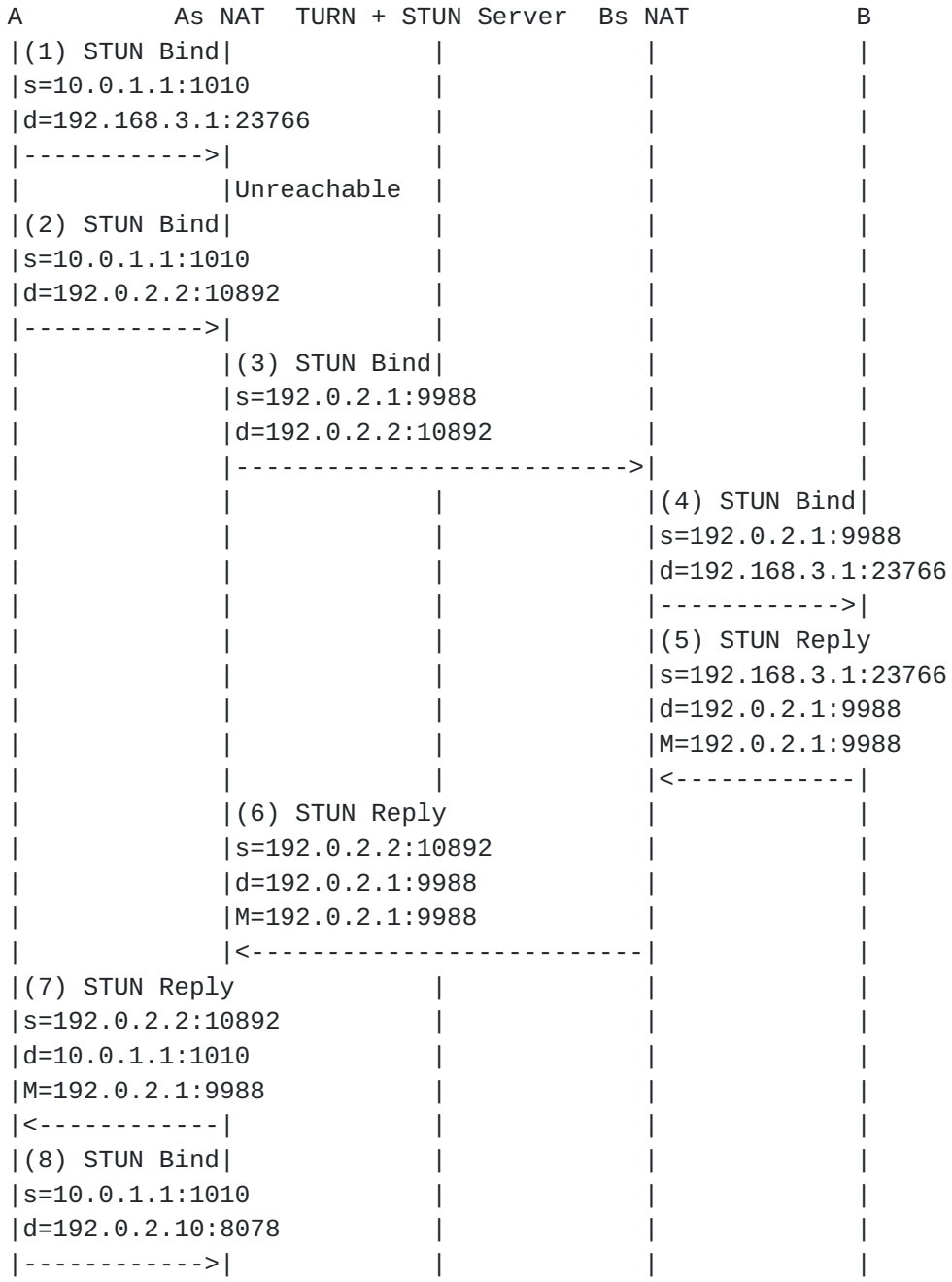
```
v=0
o=bob 2890844730 289084871 IN IP4 host2.example.com
s=
c=IN IP4 192.0.2.10
t=0 0
m=audio 8078 RTP/AVP 0
a=alt:4 1.0 : peer as88jl 192.168.3.1 23766
a=alt:5 0.8 : peer1 as88kl 192.0.2.2 10892
a=alt:6 0.4 : peer2 as88ll 192.0.2.10 8078
a=alt:7 0.4 3 peer3 as88ml 192.0.2.10 5556
```

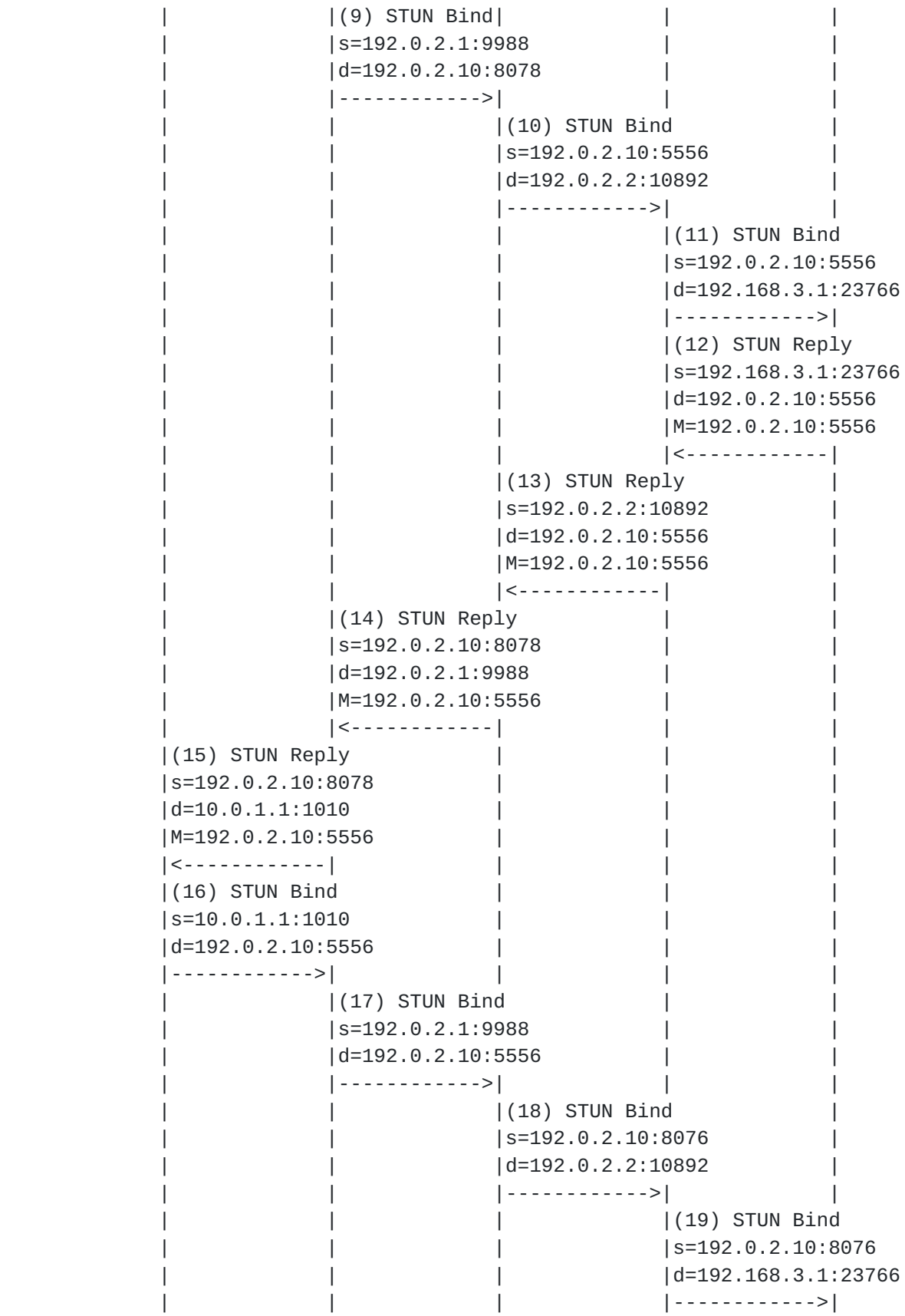
Figure 6: B's Answer

Note how the alternative with ID 7 indicates that it was derived from

the alternate with ID 3. Also, note that the four alternates use different IDs than the ones from the offer. This is for readability purposes only. The IDs are scoped to that specific agent, and there is no requirement that they do not use the same values.

This answer is sent to A. At the same time, B can send audio to A using the highest priority alternate that connectivity was established to. That is the alternate with ID 2, A's STUN derived transport address.




(9) STUN Bind
s=192.0.2.1:9988
d=192.0.2.10:8078
(10) STUN Bind
s=192.0.2.10:5556
d=192.0.2.2:10892
(11) STUN Bind
s=192.0.2.10:5556
d=192.168.3.1:23766
(12) STUN Reply
s=192.168.3.1:23766
d=192.0.2.10:5556
M=192.0.2.10:5556
(13) STUN Reply
s=192.0.2.2:10892
d=192.0.2.10:5556
M=192.0.2.10:5556
(14) STUN Reply
s=192.0.2.10:8078
d=192.0.2.1:9988
M=192.0.2.10:5556
(15) STUN Reply
s=192.0.2.10:8078
d=10.0.1.1:1010
M=192.0.2.10:5556
(16) STUN Bind
s=10.0.1.1:1010
d=192.0.2.10:5556
(17) STUN Bind
s=192.0.2.1:9988
d=192.0.2.10:5556
(18) STUN Bind
s=192.0.2.10:8076
d=192.0.2.2:10892
(19) STUN Bind
s=192.0.2.10:8076
d=192.168.3.1:23766

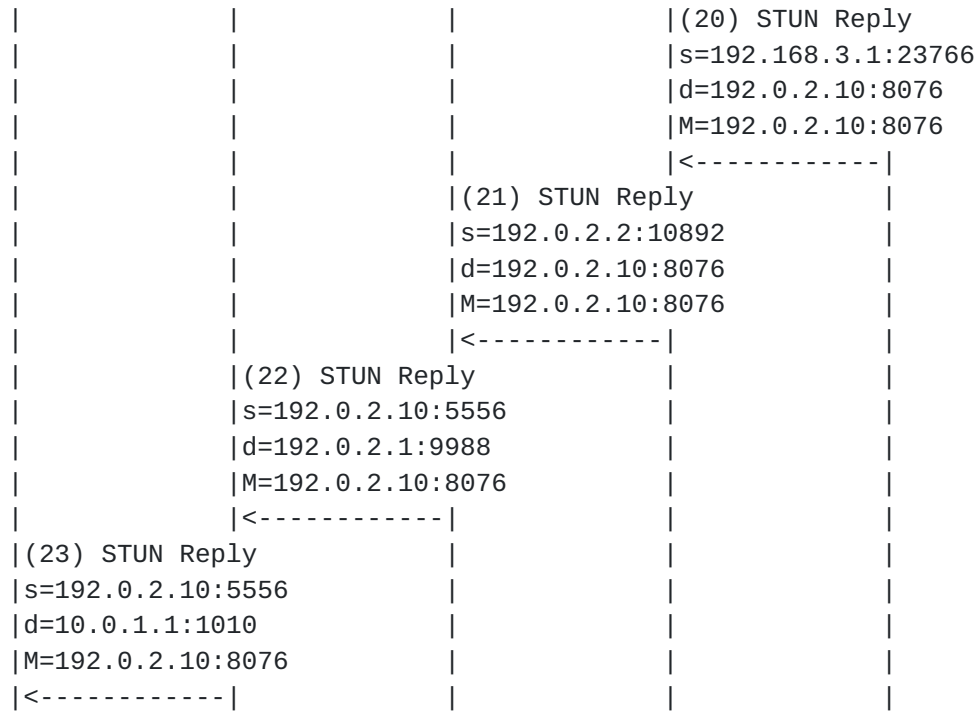


Figure 7: A's Connectivity Checks

When the answer arrives at A, A performs similar connectivity checks, shown in Figure 7. Each connectivity check is a STUN request sent from its local transport address (10.0.1.1:1010). The first is to the alternate with ID 4, which is 192.168.3.1:23766. The STUN request to this address (message 1) fails, since this is an unreachable private address. The second check is to the alternate with ID 5 (192.0.2.2:10892), which is the public address for B obtained as a result of STUN requests to the network server. Messages 2-7 represent the flow for this case. It is similar to the sequence in Figure 5 messages 2-7, differing only in the IP addresses. The result of this check provides a peer derived transport address of 192.0.2.1:9988. A already knows this address. The third connectivity check is to the alternate with ID 6 (192.0.2.10:8078). This represents A's TURN derived transport address. Messages 8-15 represent the check for this address, and they are also similar to messages 8-15 of Figure 5. This check provides A with a peer derived transport address of 192.0.2.10:5556. This represents a new address for A. It has a priority equal to the address it was derived from, which is 0.4.

The final connectivity check is to the alternate with ID 7 (192.0.2.10 5556). The SDP indicates that this address itself is a peer derived transport address. It was derived from A's transport address with ID 3, which is 192.0.2.10:8076, its TURN derived transport address. Because of that, the STUN request is sent from

the local transport address that 192.0.2.10:8076 was derived from. This local address is 10.0.1.1:1010. The message sequence for this check is represented by messages 16-23 of Figure 7. The STUN request is sent with a source address of 10.0.1.1:1010, to 192.0.2.10:5556. This is the well-known address of the TURN relay. This message passes through the NAT, and the source address is translated to A's public address, 192.0.2.1:9988 (message 17). Note that this same public address is used for all requests sent from 10.0.1.1:1010 because the NAT is full-cone. This arrives at the TURN server. The TURN server associates this message (which is just an arbitrary UDP packet as far as the TURN server is concerned) with the binding created for A. The peer in this case has been locked down. So, the packet is forwarded with a source address equal to the binding allocated to A (192.0.2.10:8076) and a destination address equal to the locked-down address (192.0.2.2:10892) (message 18). This arrives at B's NAT, where the destination address is translated to B's private address, 192.168.3.1:23766 (message 19). This arrives at B, which notes the source address in the STUN reply (192.0.2.10:8076). This reply is forwarded back to A (messages 20-23). From this, A sees a peer derived transport address of 192.0.2.10:8076. However, it already knows this address.

The result of the connectivity checks is that A determines it has connectivity to the alternates with IDs 5, 6 and 7. Of these, the one with ID 5 has the highest priority, and so this one is used to send media. Of course, A could have been sending media to B during these tests using the address in the m and c lines, which represents B's TURN derived transport address. Once the connectivity checks complete, A can switch to the one with ID 5, which is B's STUN derived transport address.

The connectivity checks also provided A with a new peer derived transport address - 192.0.2.10:5556 - with a priority of 0.4. However, A had received STUN requests on its alternates with IDs 2 and 3. The one with ID 2 (its STUN derived transport address) has higher priority than 0.4. So, A knows that generating a new ICE cycle to convey this address would not be useful. Thus, no new offer is sent. Indeed, since A had received a STUN request from B on its STUN derived transport address, A knows that its lower priority derived transport address is no longer needed. So, it is able to free up the TURN derived transport address a few seconds later. The same goes for B. Once it receives the STUN request to its TURN derived transport address (message 11 of Figure 7, it can free its TURN derived transport address.

In conclusion, the result in this case is that A and B will communicate with each other using their STUN derived transport addresses.

2.2 Symmetric NAT

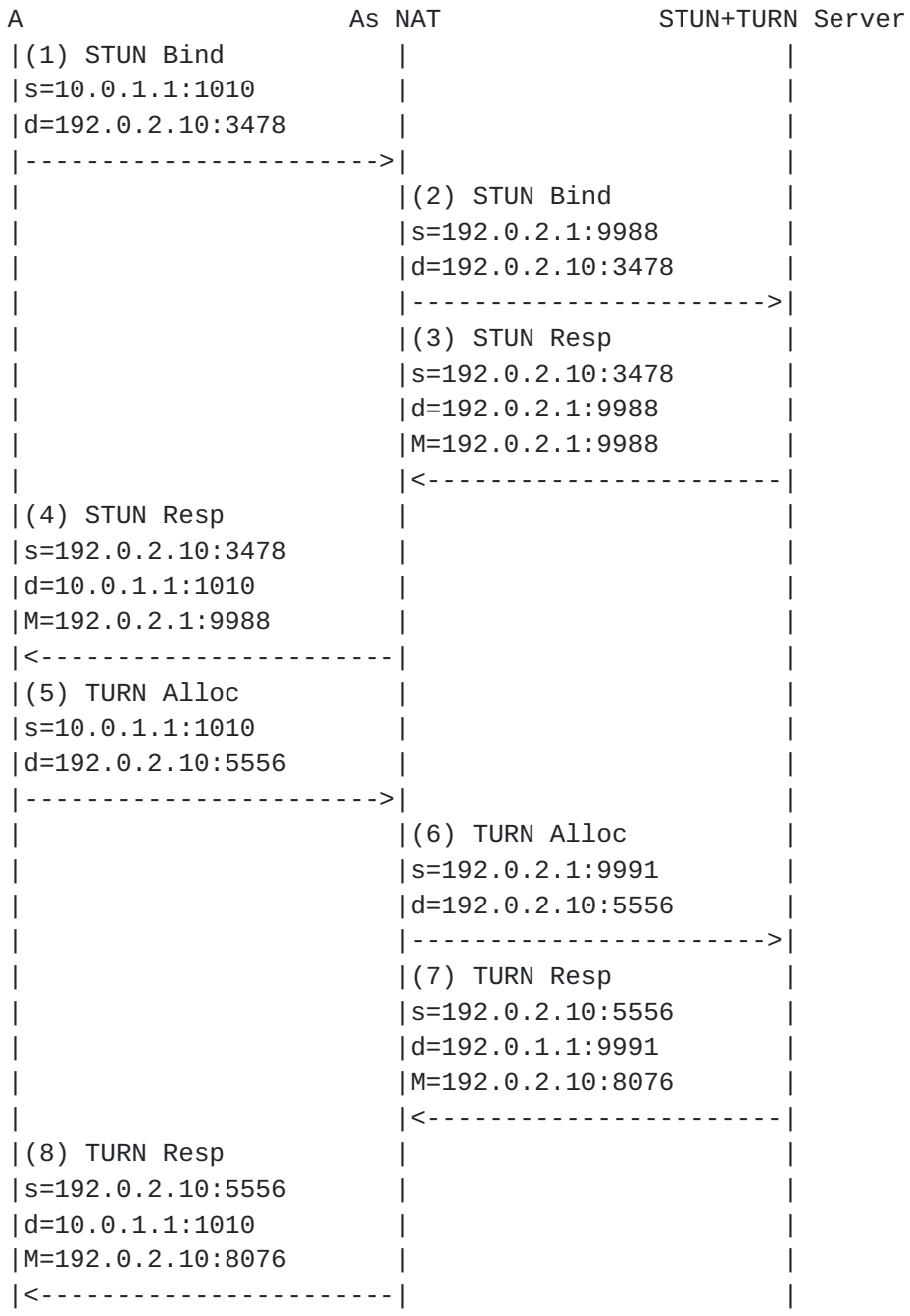


Figure 8: A's Unilateral Allocations

In this case, both residential users have symmetric NATs. The call starts again with A performing its unilateral allocations, as is shown in Figure 8. This message sequence is nearly identical to that of Figure 2. The only difference is that, because the NAT is

symmetric, different bindings are allocated for the two STUN and two TURN queries. A's discovers an identical set of addresses, however, and so generates the same offer as in Figure 3.

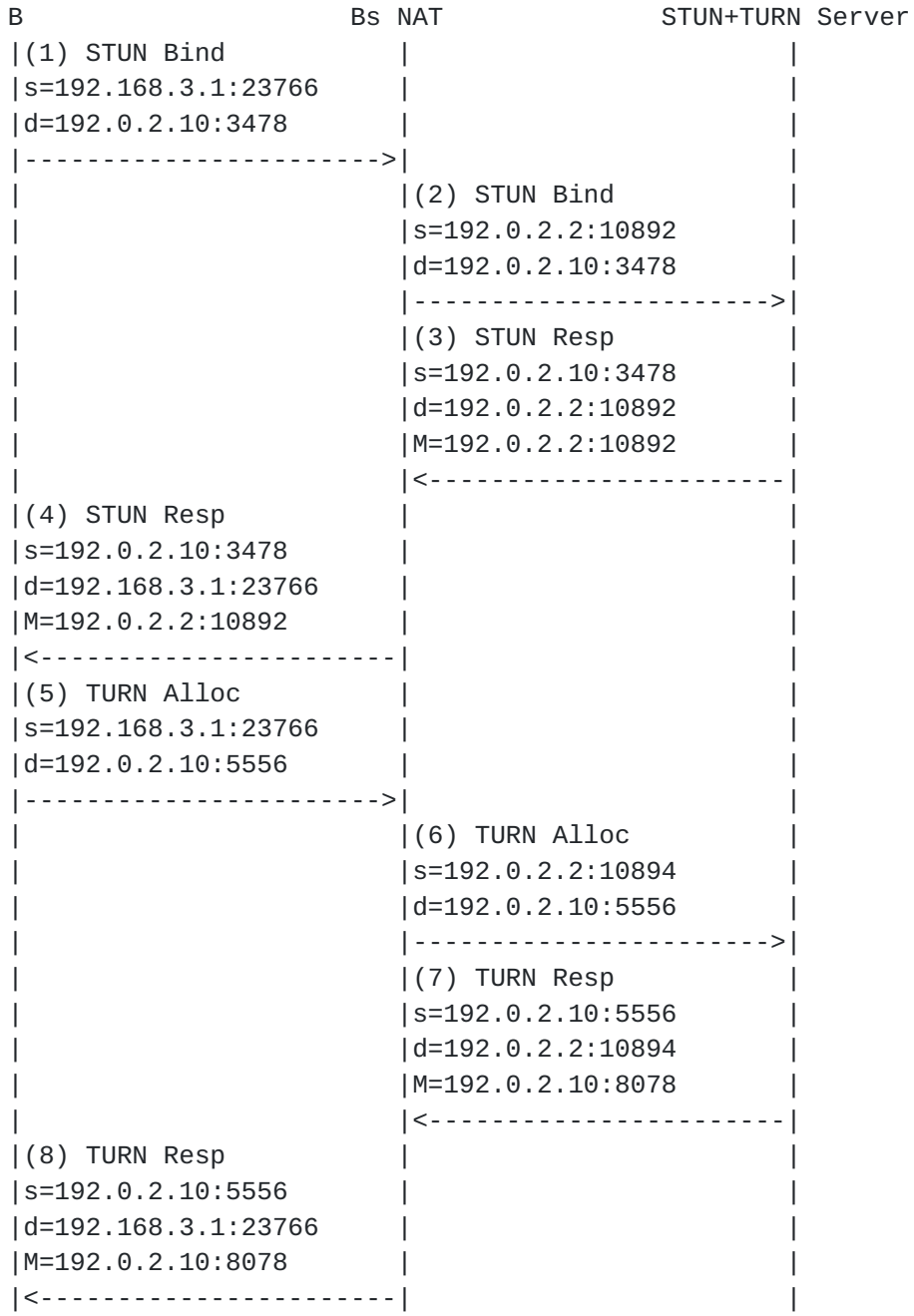
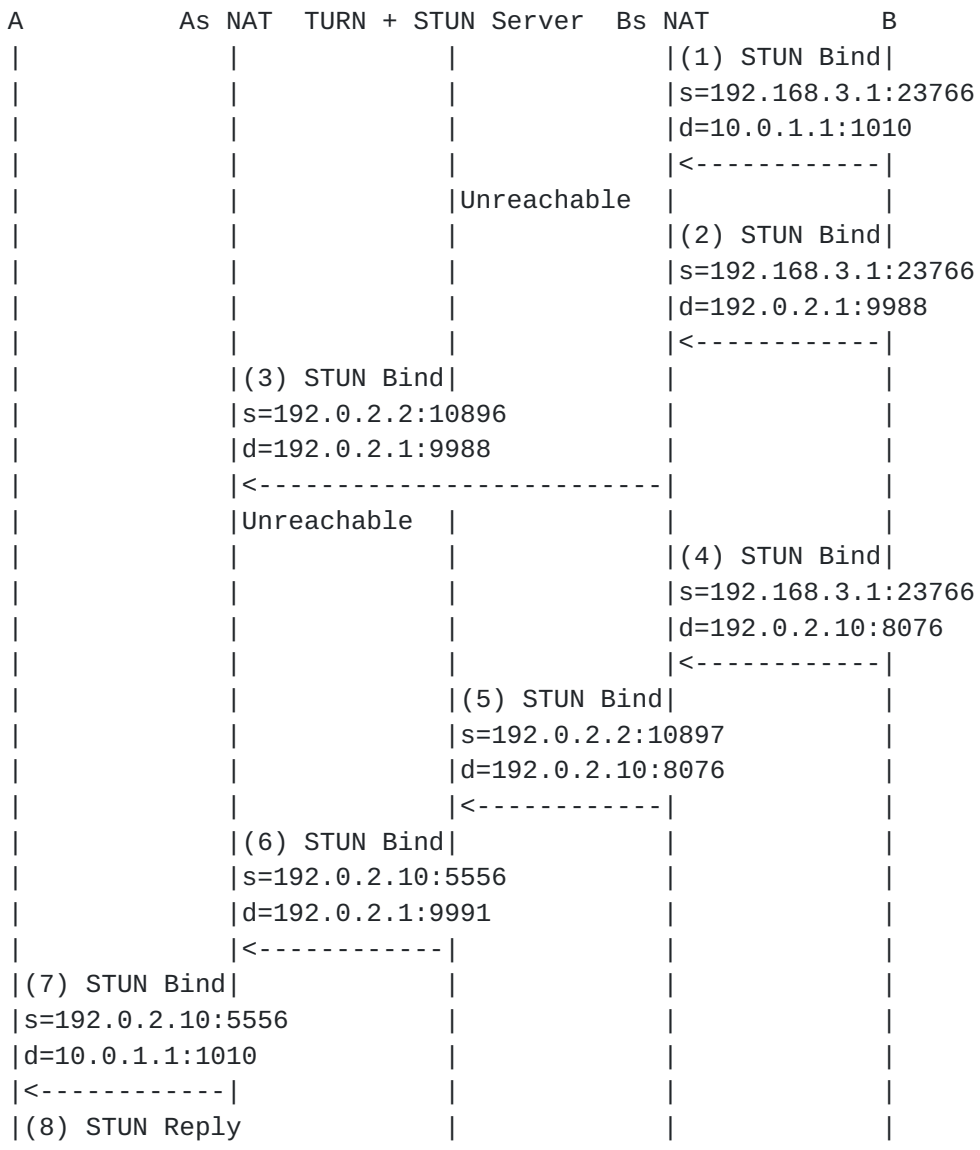


Figure 9: B's Unilateral Allocations

When B receives this offer, it performs its unilateral allocations. Like A's, these allocations (shown in Figure 9) are almost identical

to those in Figure 4. They differ in the same way - the NAT will allocate a different binding for each of the two STUN and two TURN queries. However, the set of derived transport address is the same. B now begins performing connectivity checks. These are shown in Figure 10. As in the previous case (Figure 5), the STUN request to 10.0.1.1:1010 fails. However, here, the STUN request to 192.0.2.1:9988 also fails. Thats because this packet arrives at A's NAT, and the NAT finds that the public transport address 192.0.2.1:9988 has been allocated, however, it was allocated when the client sent to 192.0.2.10:3478. Here, the source address is not 192.0.2.10:3478, and so the packet is discarded. The STUN request to 192.0.2.10:8076 does work, however. Thats because the TURN server sends the request from the same IP address and port that it received the original TURN allocation request on.



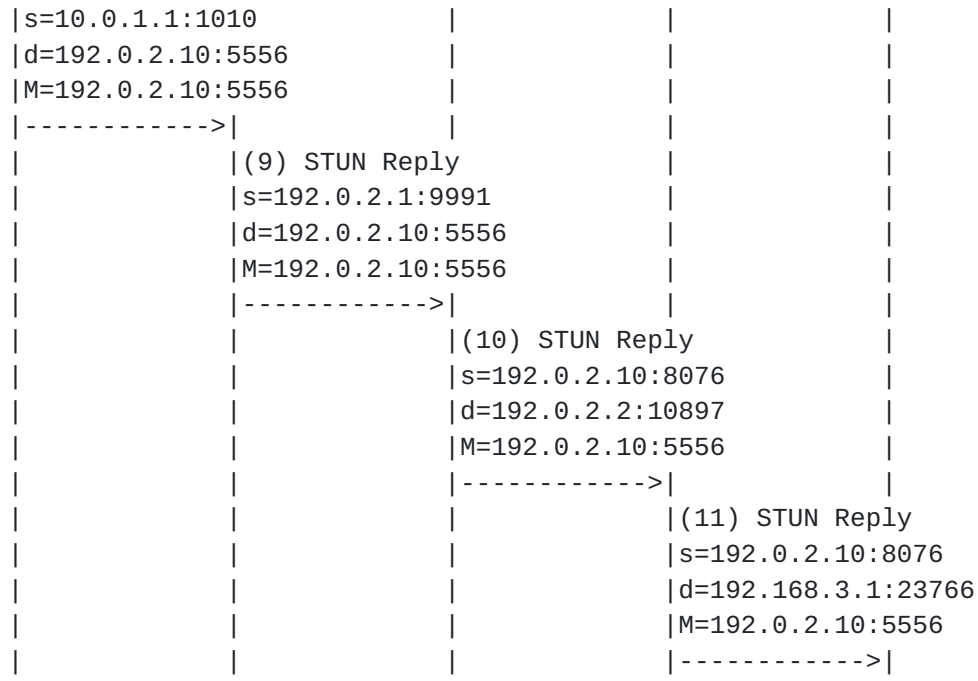
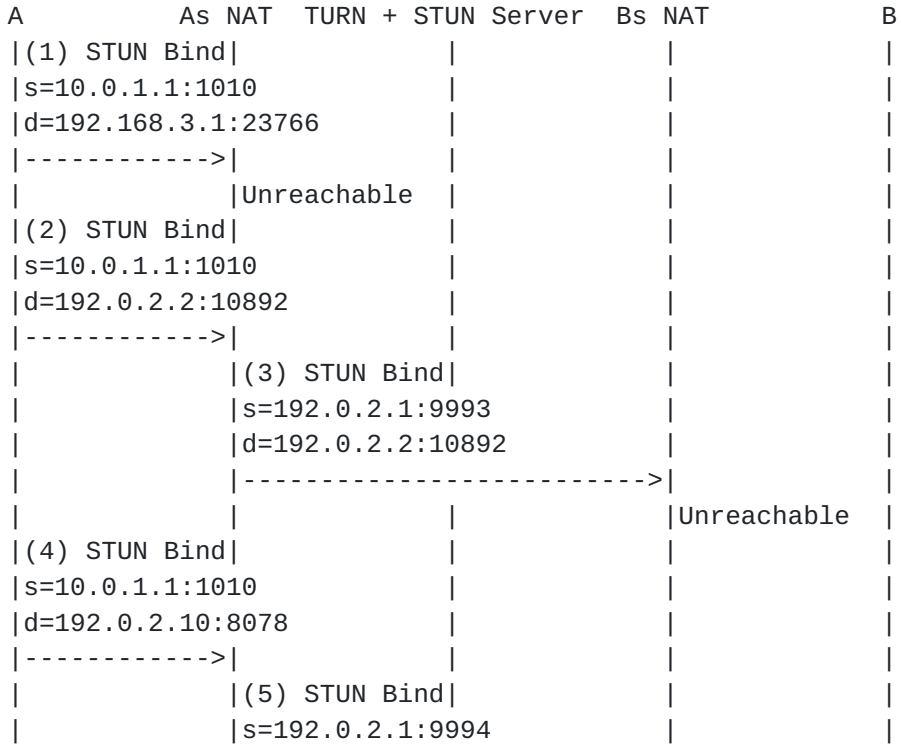
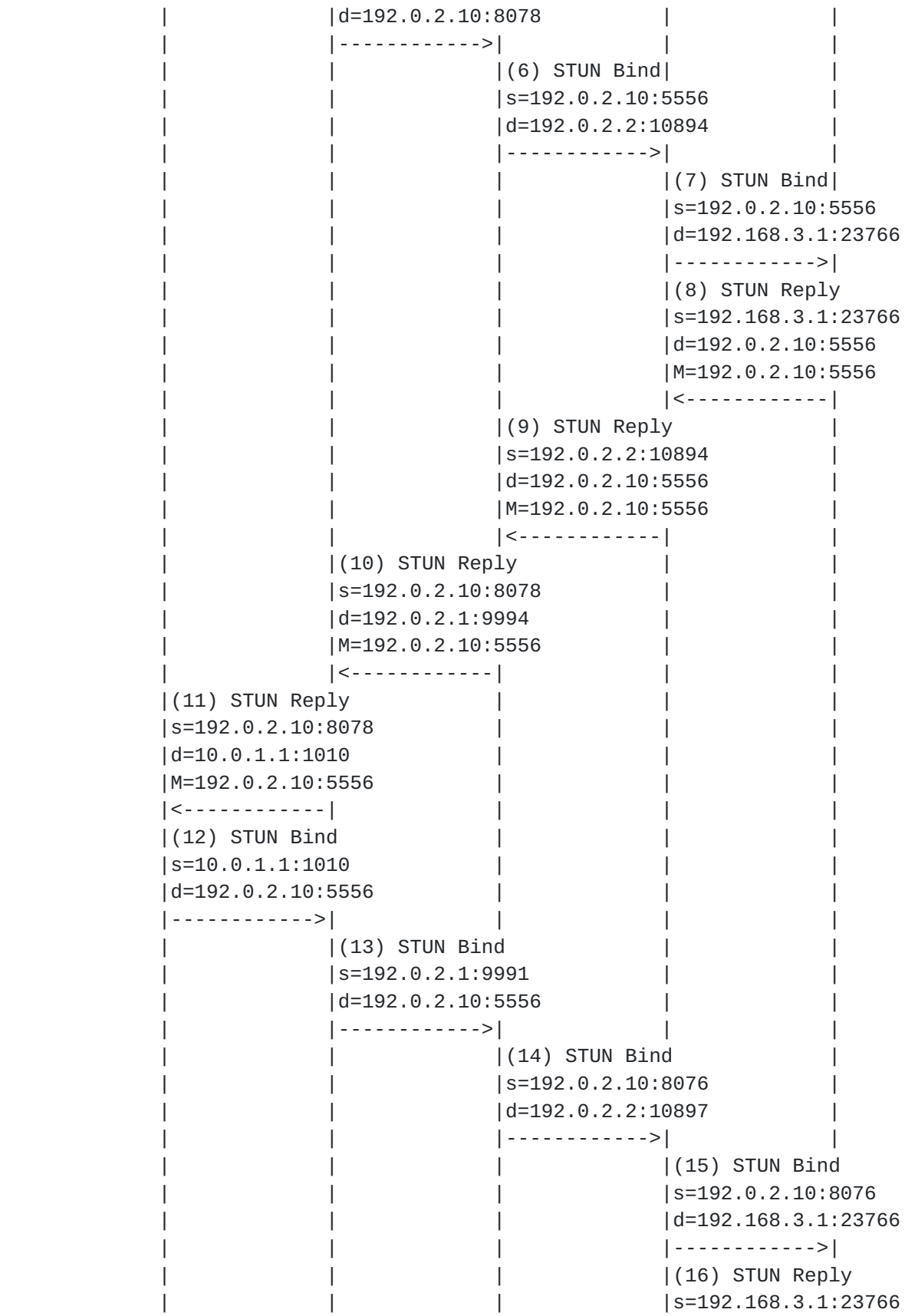


Figure 10: B's Connectivity Checks

B's answer to A is the same as in Figure 6. However, B has only established connectivity to A's TURN derived transport address, and so it sends media there.




d=192.0.2.10:8078
(6) STUN Bind
s=192.0.2.10:5556
d=192.0.2.2:10894
(7) STUN Bind
s=192.0.2.10:5556
d=192.168.3.1:23766
(8) STUN Reply
s=192.168.3.1:23766
d=192.0.2.10:5556
M=192.0.2.10:5556
(9) STUN Reply
s=192.0.2.2:10894
d=192.0.2.10:5556
M=192.0.2.10:5556
(10) STUN Reply
s=192.0.2.10:8078
d=192.0.2.1:9994
M=192.0.2.10:5556
(11) STUN Reply
s=192.0.2.10:8078
d=10.0.1.1:1010
M=192.0.2.10:5556
(12) STUN Bind
s=10.0.1.1:1010
d=192.0.2.10:5556
(13) STUN Bind
s=192.0.2.1:9991
d=192.0.2.10:5556
(14) STUN Bind
s=192.0.2.10:8076
d=192.0.2.2:10897
(15) STUN Bind
s=192.0.2.10:8076
d=192.168.3.1:23766
(16) STUN Reply
s=192.168.3.1:23766

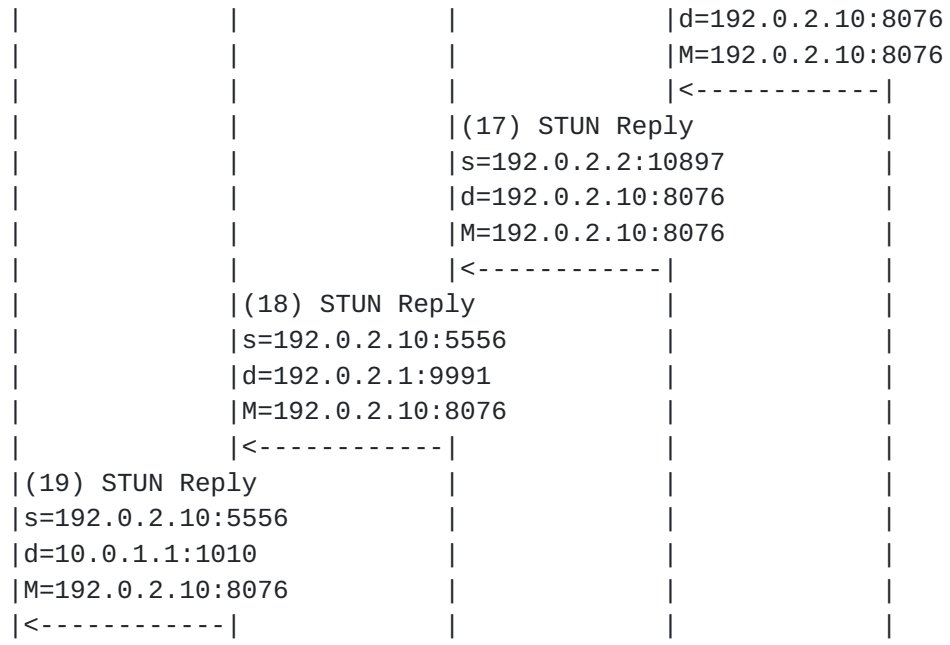


Figure 11: A's Connectivity Checks

When A gets the answer, it too performs its connectivity checks, as shown in Figure 11. As expected, the connectivity checks to B's private address and its STUN derived transport addresses fail. The checks to B's TURN derived transport address succeeds, as does the check to B's peer derived transport address. Both have a qvalue of 0.4. However, a peer-derived address is always preferred. So, A will send media to B using 192.0.2.10:5556, which will reach B as a result of the lock-down on its own TURN binding. As in the full-cone case, A won't bother to perform another offer with the new peer derived transport address it learned from message 19 (192.0.2.10:5556), since it knows that this is not of higher priority than ones that B has already connected to.

Once A connects to B's peer derived address (messages 12 to 19 in Figure 11), B knows that its equal priority TURN derived transport address won't be used, so it can free it.

OPEN ISSUE: The same argument can be made about A, in which case both sides would free their TURN addresses, and nothing works. Need to come up with a sane prioritization so it doesnt happen.

3. Basic Enterprise

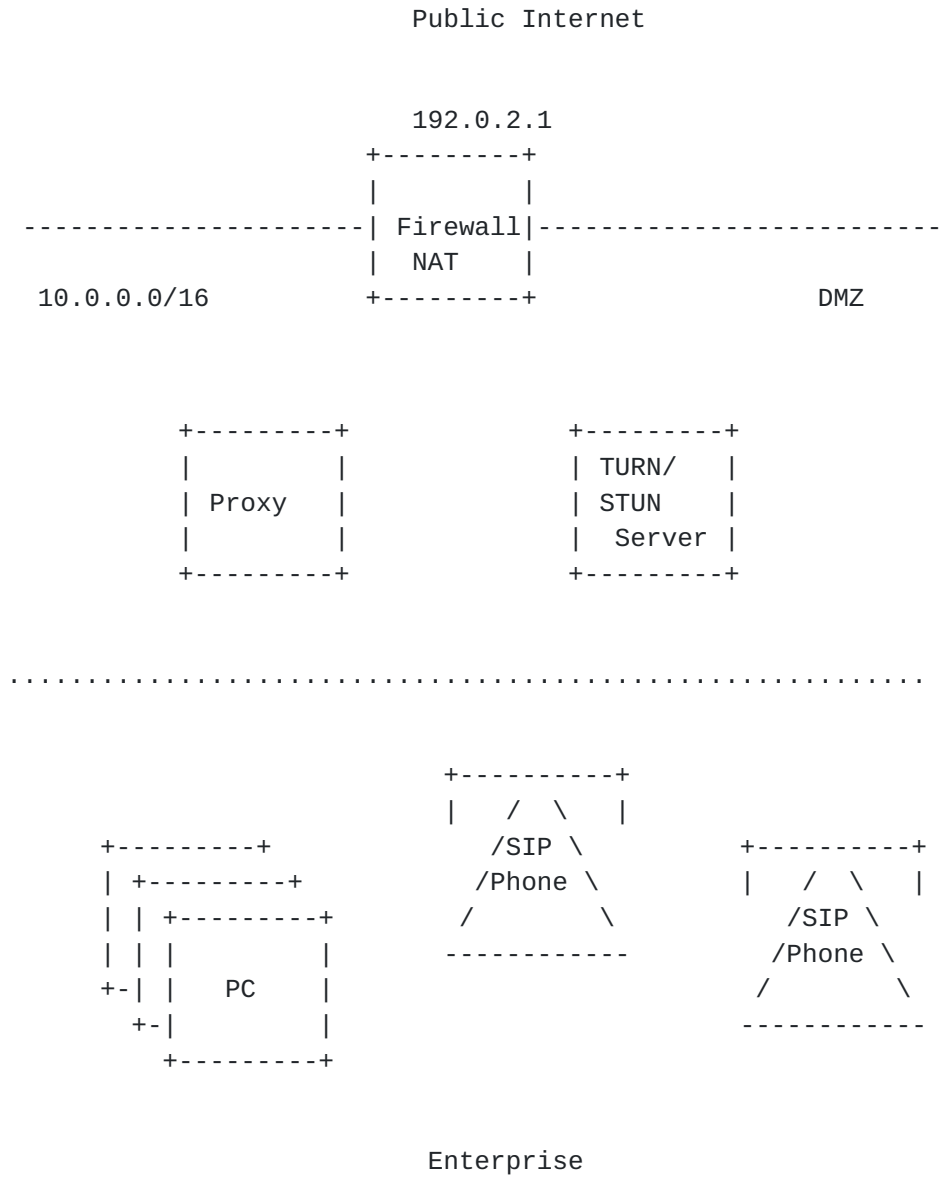


Figure 12: Enterprise Configuration

In this scenario, shown in Figure 12 there is an enterprise that wishes to deploy VoIP. The enterprise has a single site, and there is a firewall/NAT at the border to the public Internet. This NAT is symmetric. Internally, the enterprise is using 10.0.0.0/16. Behind the firewall, within the DMZ, is a TURN/STUN server and a SIP proxy. The firewall has been configured to allow incoming traffic to port 5060 to go to the SIP proxy. It has also allowed incoming UDP traffic on a specific port range to go to the TURN/STUN server. The

TURN server has an internal address of 10.0.1.10. This port range contains enough addresses to allow simultaneous conversations to cover the needs of the enterprise, but no more. External traffic sent to 192.0.2.1:8000 to 192.0.2.1:9000 is port forwarded to 10.0.1.10:8000 to 10.0.1.10:9000, respectively. That range is configured on the TURN/STUN server, so that the TURN server allocates addresses within this range.

Within the enterprise, PCs and hardphones are used for VoIP. All of them are configured to use the proxy and TURN/STUN server that is run by the enterprise. Furthermore, all of them are configured to use the TURN SEND mechanism for doing connectivity checks.

All call flows in this section only indicate RTP. The flows for RTCP are not shown.

3.1 Intra-Enterprise Call

In this section, we consider calls between two users in the same enterprise.

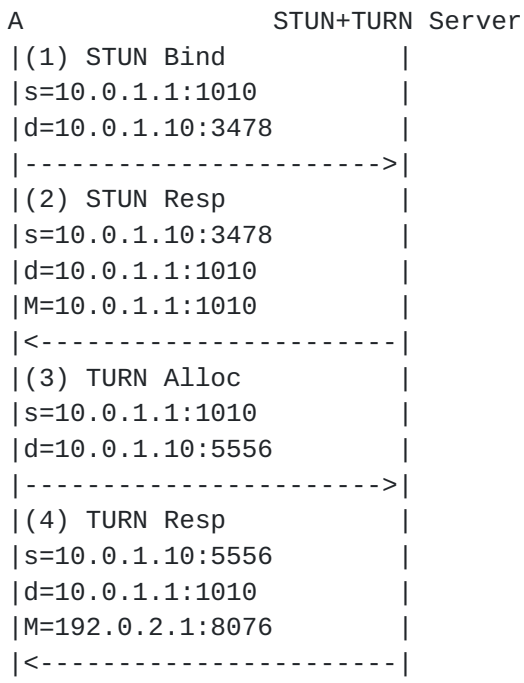


Figure 13: A's Unilateral Allocations

First, user A performs its unilateral allocations. This is shown in Figure 13. The STUN allocation does not yield a new address, but the TURN allocation, of course, does. The TURN address is publically routable. As a result, the offer from A to B has two addresses, as

shown in Figure 14.

```
v=0
o=alice 2890844730 2890844731 IN IP4 host.example.com
s=
c=IN IP4 192.0.2.1
t=0 0
m=audio 8076 RTP/AVP 0
a=alt:1 1.0 : user 9kksj== 10.0.1.1 1010
a=alt:2 0.5 : user1 9kksk== 192.0.2.1 8076
```

Figure 14: A's Offer

B receives this offer. It performs its own unilateral allocations, shown in Figure 15.

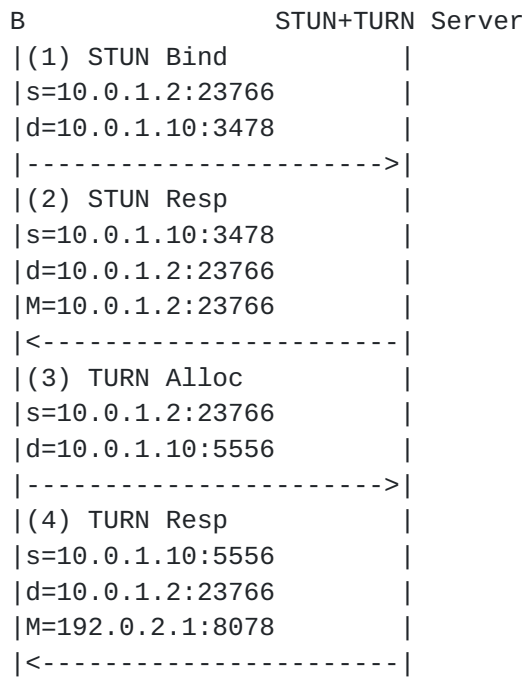
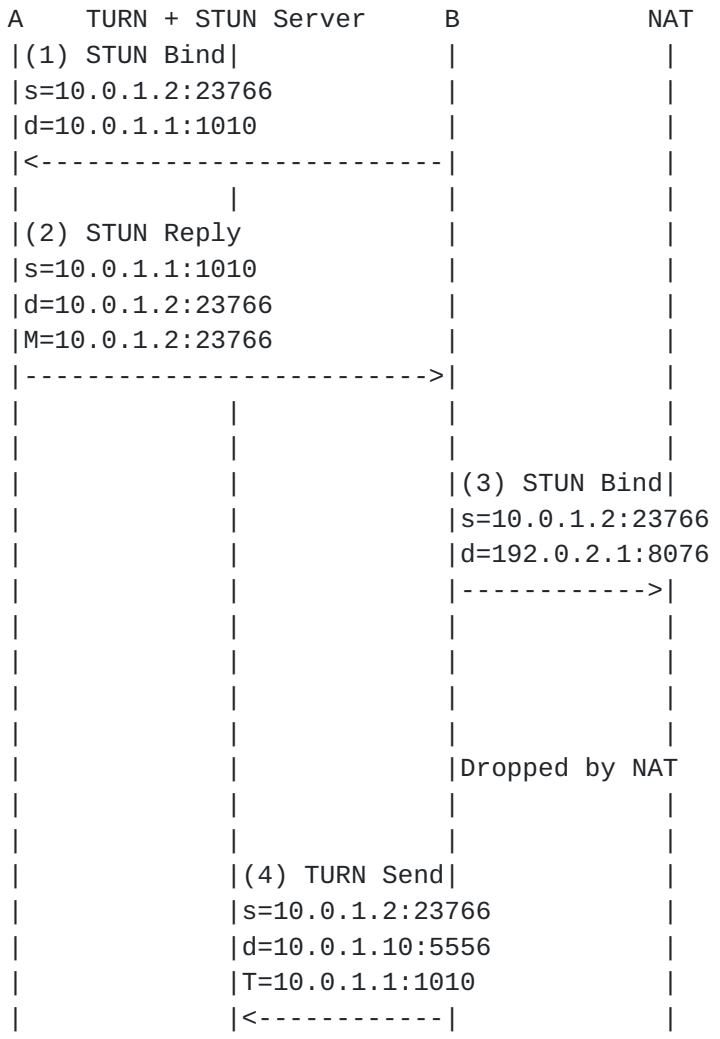


Figure 15: B's Unilateral Allocations

The STUN derived transport address equals its local transport address, so no additional addresses are obtained through that route. TURN provided B with a public address. Next, B performs connectivity checks against the two alternatives provided by A. These checks are shown in Figure 16. The connectivity check to the alternate with ID 1, A's local transport address, succeeds, since both users are within the same address realm. The connectivity to check to the alternate with ID 2, which is the TURN server address on the public Internet,

fails. This is because the NAT does not support receipt of requests from internal hosts that are targeted towards internal bindings. As a result, the STUN request is dropped by the NAT.

Because of its configuration, B also attempts to perform connectivity checks by sending STUN Bind requests though its TURN relay, using the TURN SEND command. As described in ICE, these connectivity checks need to be performed sequentially, not in parallel. B first attempts a send to deliver a STUN Bind request to A's local transport address (message 4). This is relayed by the TURN server to A, using the internal version of B's TURN derived transport address (10.0.1.10:8078) as the source address (message 5). This is the address that the NAT will translate 192.0.2.2:8078 into when it receives packets externally. A replies to this (message 6), reporting to B a new address, 10.0.1.10:8078. This is received by the TURN server, causing lock down to occur. The TURN server forwards this response back to B.



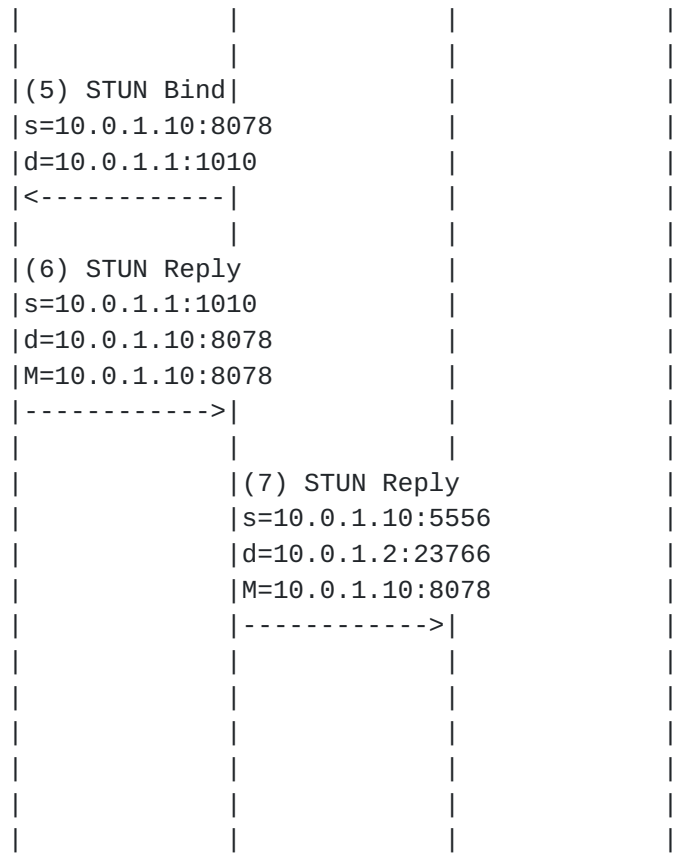


Figure 16: B's Connectivity Test

Based on this, B generates the answer shown in Figure 17. Since B has established connectivity to A's local transport address, it begins sending media there.

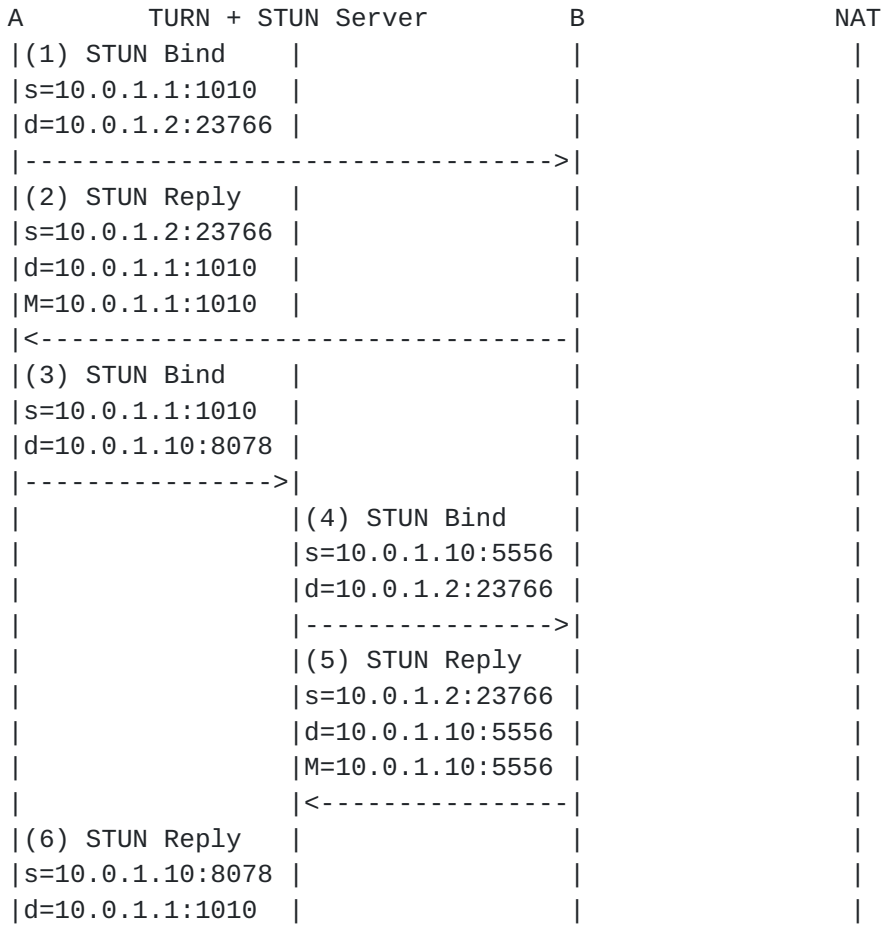
```
v=0
o=bob 2890844730 289084871 IN IP4 host2.example.com
s=
c=IN IP4 192.0.2.1
t=0 0
m=audio 8078 RTP/AVP 0
a=alt:4 1.0 : peer as88jl 10.0.1.2 23766
a=alt:6 0.5 1 peer2 asjj8n 10.0.1.10 8078
a=alt:5 0.5 : peer1 as88kl 192.0.2.1 8078
```

Figure 17: B's Answer

Now, A performs its connectivity checks, shown in Figure 18. First, it checks for connectivity to B's local transport address (message 1). This connectivity check passes, and does not provide A with a

new address (message 2). Next, A checks for connectivity to 10.0.1.10:8078, the internal version of B's TURN derived transport address. This connectivity check (messages 3-6) also succeed, and provide A with a new peer derived transport address (10.0.1.10:5556). However, this address would have a lower priority (0.5) than that of one that B has already connected to (A's local transport address), and so A does not bother with another ICE cycle. The check to B's public TURN derived transport address fails (message 7). Since A discovers connectivity to a high priority transport address, it does not bother to perform its connectivity checks by relaying STUN messages through its TURN server. Both A and B can now free their TURN derived addresses, since both have established connectivity to higher priority addresses. The call proceeds with media flowing directly between A and B, as desired.

Note, however, that this call flow would not have worked if A supported ICE, but B didn't. Thats because the default TURN address will not work for internal clients. In enterprises where this is a concern, an alternate deployment, described in Section 4, works properly.



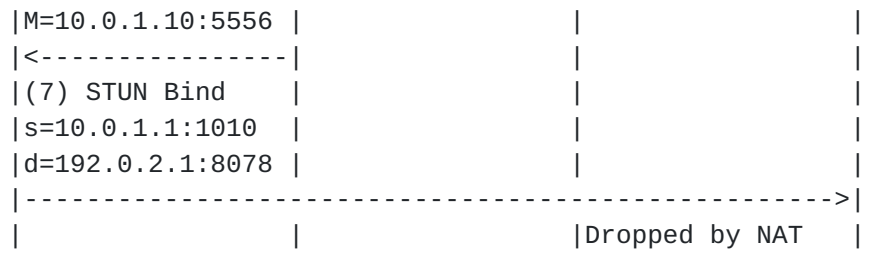


Figure 18: A's Connectivity Checks

3.2 Extra-Enterprise Call

In this case, user A within the enterprise calls some user B, not within the enterprise. B is connected to the Internet through a PSTN gateway, and as a result, appears as a UA on the public Internet. Presumably this is some gateway run by a third party termination provider that is being used by the enterprise. Furthermore, this gateway does not support ICE at all, and so will ignore the alt parameters in the SDP.

First, A performs its unilateral allocations. This proceeds identically as shown in Figure 13. It generates the same offer as shown in Figure 14. This gets routed to the called party on the public Internet. This party generates an answer. However, since the called party does not support ICE, the answer has no alt attributes. It has a single IP address and port listed in the c and m lines. As a result, the caller, A, needs to send media there. However, the enterprise policy prohibits outbound UDP traffic from end user devices. Thus, A has been configured to ensure outbound media flows through the TURN server. ICE would normally discover this, and media would flow that way. However, since ICE is not supported, it needs to be done explicitly by the client.

To accomplish this, A performs another, separate unilateral allocation to obtain another TURN address. It does not advertise this address to the called party. Instead, it issues a TURN SEND command to the IP address and port in the SDP answer. This send command contains the first RTP packet to send. From that point forward, A sends its media packets to the TURN server. The TURN server will forward those packets to the last address used in a SEND command, as long as lockdown has not occurred. Here, it will not, since the address learned from the TURN server is never advertised to any peers.

3.3 Inter-Enterprise

In this case, a user in one enterprise calls a user in another enterprise. In this configuration, media needs to flow through the TURN relays run by both enterprises, since the policies of both enterprises require this. We assume that B's enterprise is using 192.168/16 internally, and it has an external public IP address of 192.0.2.2. The TURN/STUN server is running on 192.168.1.10, using port 3478 for STUN and 5556 for TURN. Packets sent to 192.0.2.2:6500 to 192.0.2.2:6600 are forwarded to 192.168.1.10:6500 to 192.168.1.10:6600 respectively.

First, A performs its allocations. These are identical to the ones in Figure 13. The offer sent by A, as a result, is identical to Figure 14.

This call is received by B. B performs its allocations, shown in Figure 19. These are similar to those of Figure 15, differing only in the addresses used.

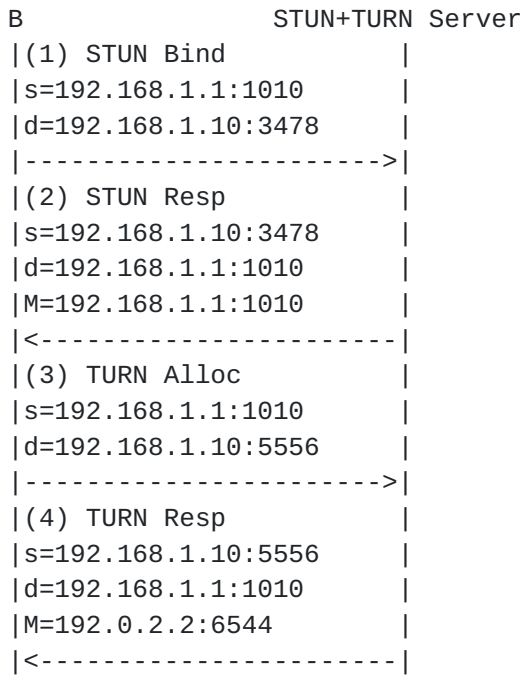
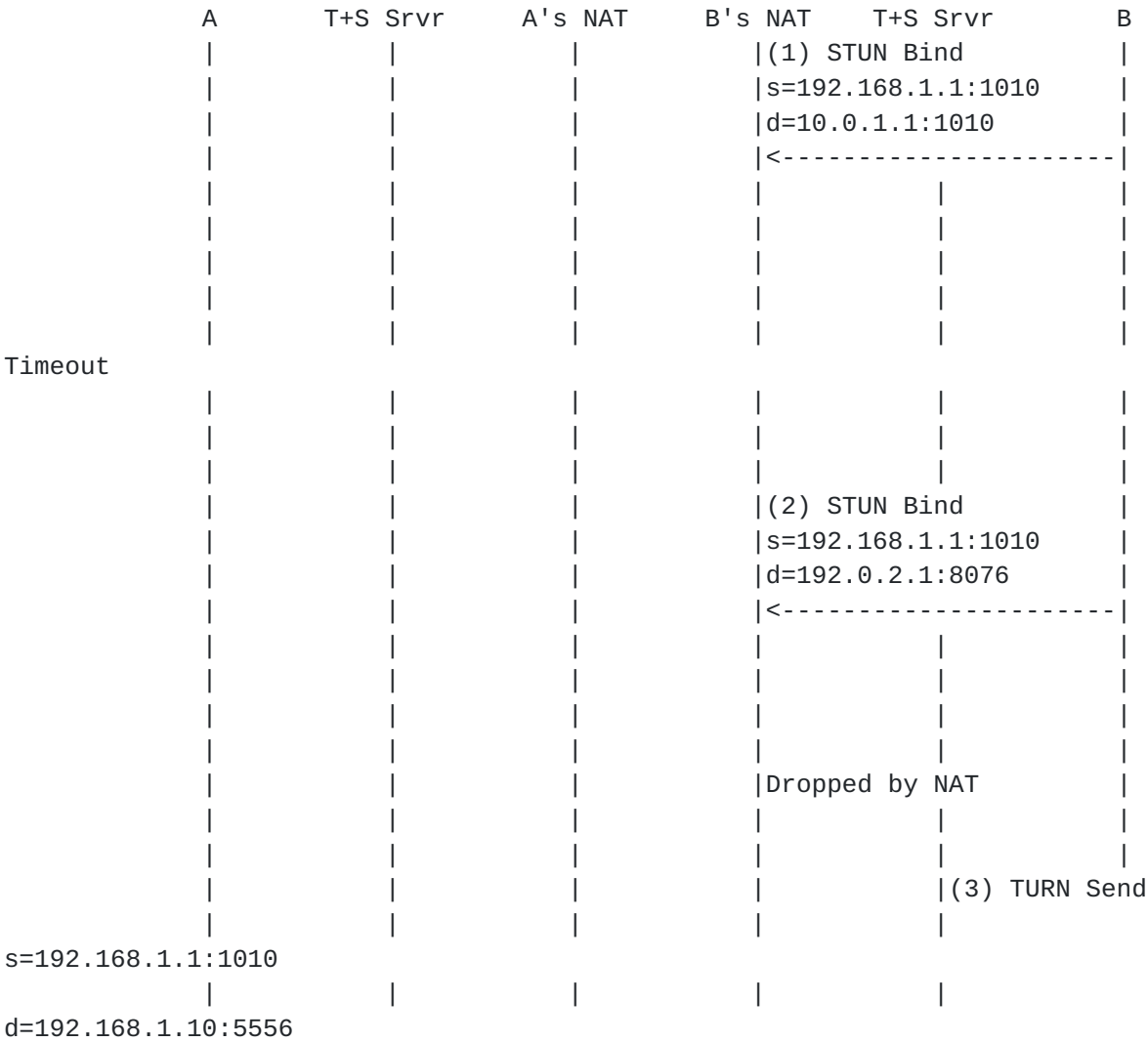


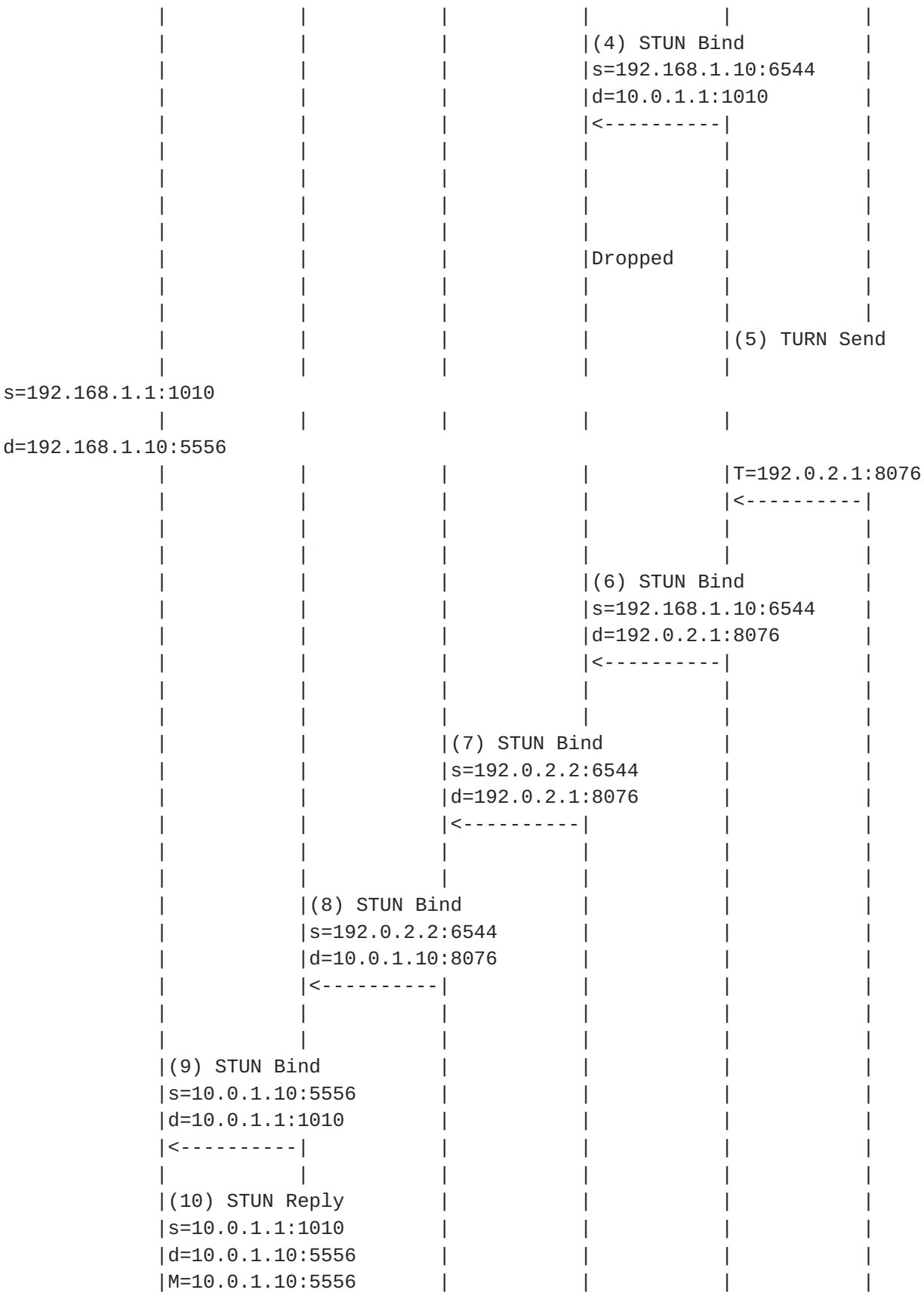
Figure 19: B's Unilateral Allocations

Next, B performs its connectivity checks, as shown Figure 20. First, B checks connectivity to A's local transport address (10.0.1.1:1010). This is unroutable within B's network, and so the STUN request is dropped by the routers in the network, and the check times out and fails. In parallel, B checks connectivity to A's TURN derived

transport address (192.0.2.1:8076). It sends a STUN Bind request to this address (message 2). This arrives at B's firewall/NAT. However, the firewall function does not allow outbound UDP packets from internal clients, and so the packet is dropped. This check also times out and fails. B also begins checking connectivity to A's two addresses by SENDing the STUN requests through its TURN server. First, B tries A's local transport address (message 3). This is relayed by the TURN server to 10.0.1.1:1010, which is dropped by the routers as well. Finally, B tries A's TURN derived transport address (message 4). This is successfully relayed all the way to A, as a result of the static bindings in place in A's and B's NATs. A sees a source address of 10.0.1.10:5556, which it reports back in the STUN reply. The STUN request (message 8) to A's TURN server locks down the binding, and the STUN reply (message 13) locks down the binding at B's TURN server. Based on the connectivity checks, B has learned a single new peer derived transport address, 10.0.1.10:5556.




T=10.0.1.1:1010


(4) STUN Bind
s=192.168.1.10:6544
d=10.0.1.1:1010
Dropped
(5) TURN Send
s=192.168.1.1:1010
d=192.168.1.10:5556
T=192.0.2.1:8076
(6) STUN Bind
s=192.168.1.10:6544
d=192.0.2.1:8076
(7) STUN Bind
s=192.0.2.2:6544
d=192.0.2.1:8076
(8) STUN Bind
s=192.0.2.2:6544
d=10.0.1.10:8076
(9) STUN Bind
s=10.0.1.10:5556
d=10.0.1.1:1010
(10) STUN Reply
s=10.0.1.1:1010
d=10.0.1.10:5556
M=10.0.1.10:5556

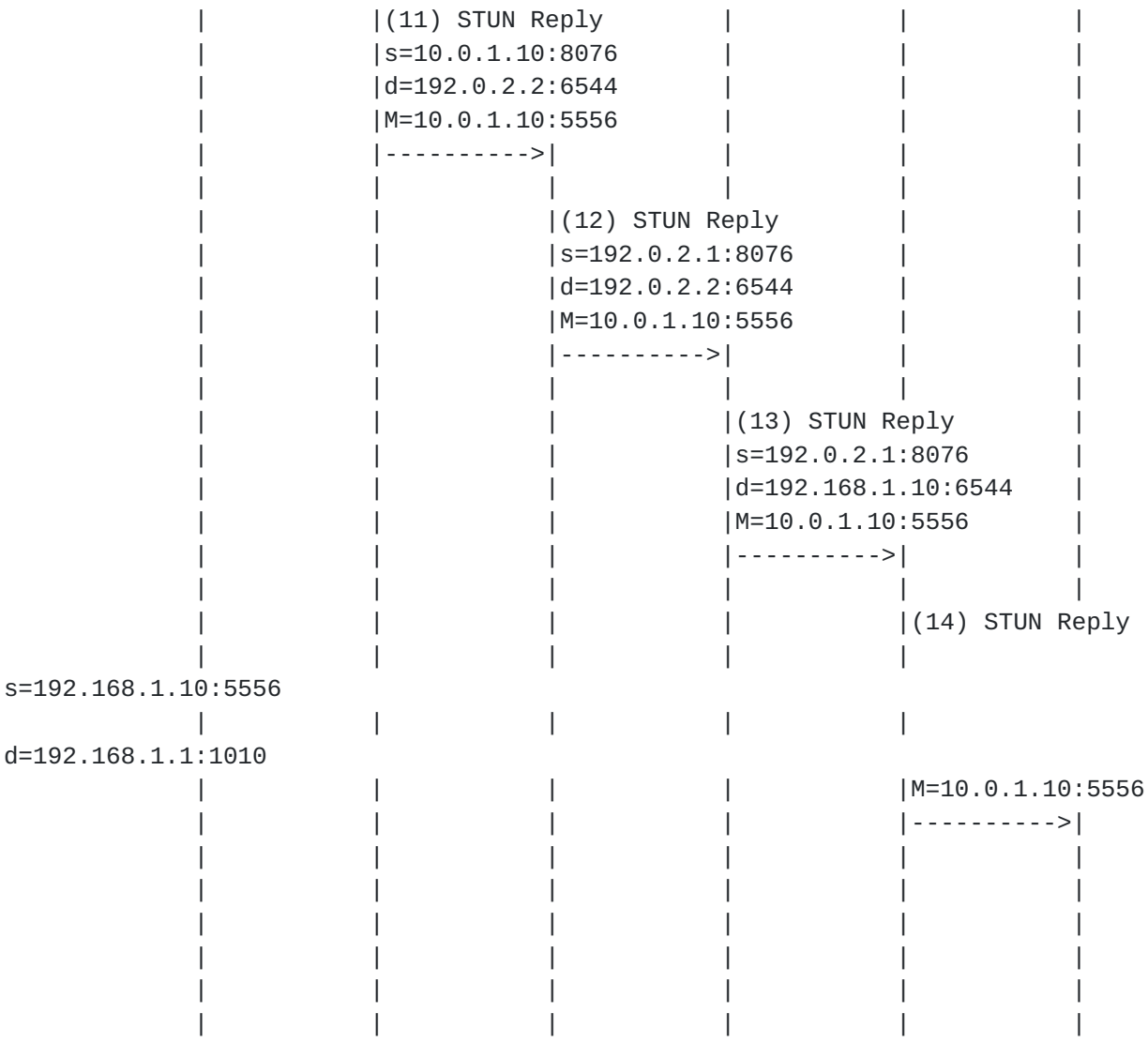
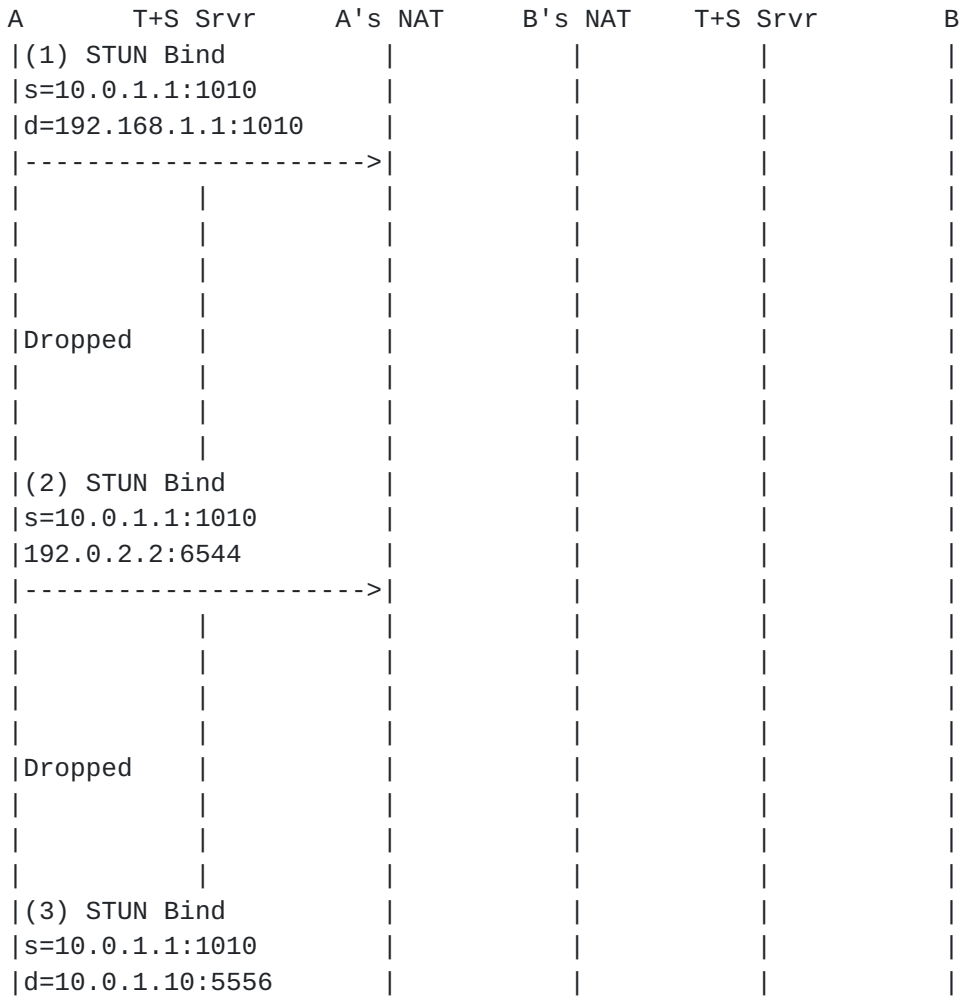


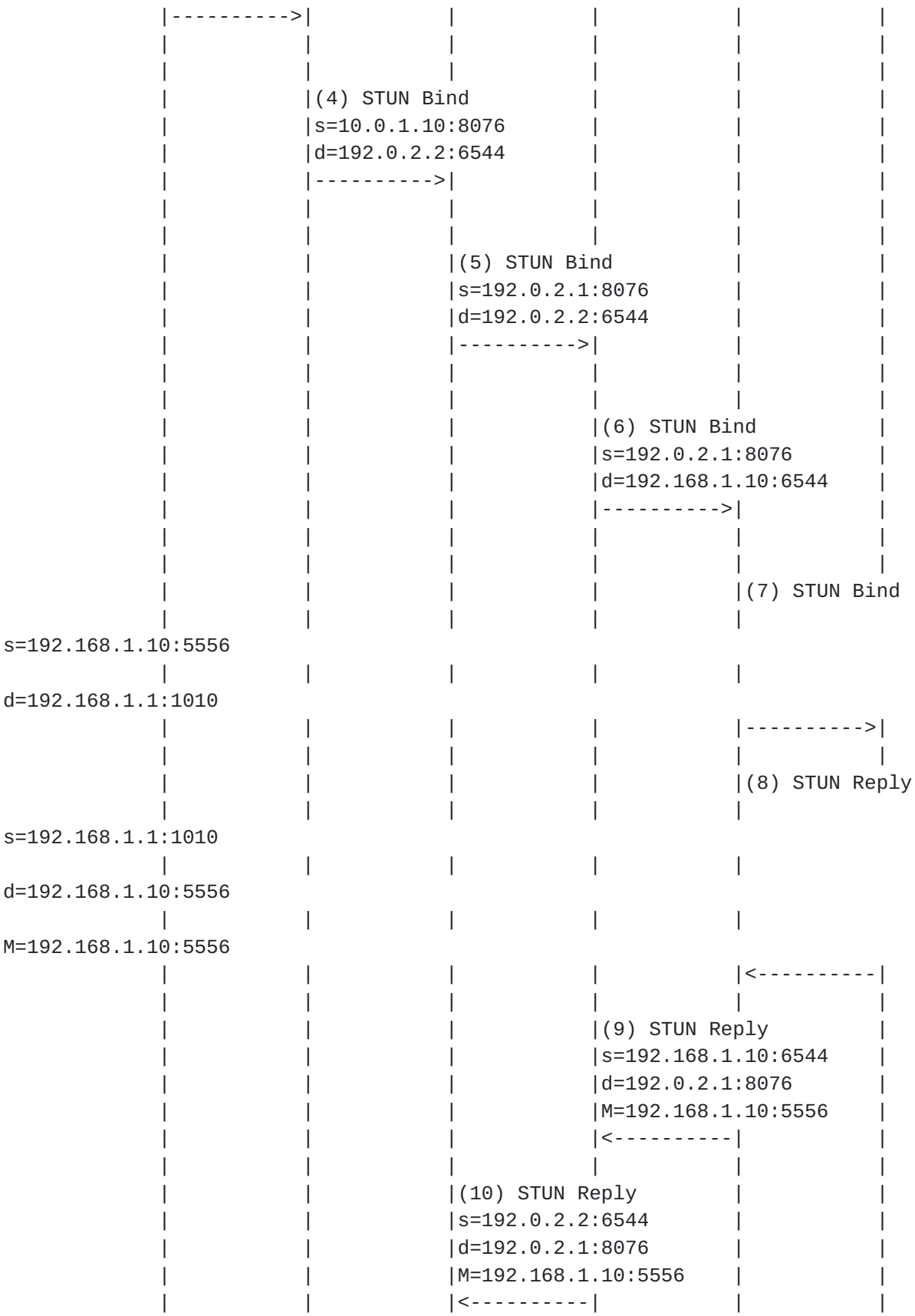
Figure 20: B's Connectivity Test

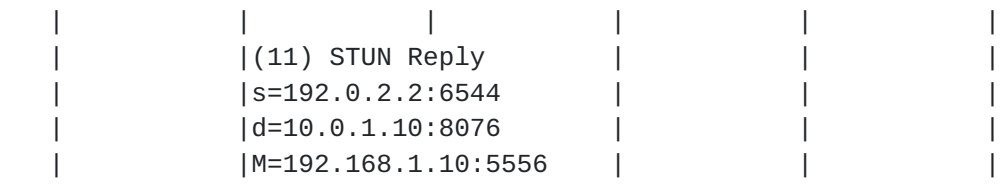
B's connectivity check showed that the only place where media can be sent is through its relay. Since the binding has been locked down, B knows it can just send raw media packets to the relay, which will be forwarded appropriately. As such, it begins sending media through the relay pairs. B also generates its answer:

```
v=0
o=bob 2890844730 289084871 IN IP4 host2.example.com
s=
c=IN IP4 192.0.2.2
t=0 0
m=audio 6544 RTP/AVP 0
a=alt:4 1.0 : peer as88jl 192.168.1.1 1010
a=alt:5 0.5 : peer1 as88kl 192.0.2.2 6544
a=alt:6 0.5 2 peer3 hh8sdl 10.0.1.10 5556
```

Now, A performs its connectivity checks, which are shown in Figure 22. These checks are similar to those of Figure 20. A's TURN server relays the STUN request towards B's TURN server because of the lock-down from B;s connectivity test. A's test reveals connectivity to 10.0.1.10:5556, which is B's peer derived address. Since connectivity was established there, A does not bother doing connectivity checks by SENDing STUN requests through its TURN server. The media proceeds to flow through both relays.




(4) STUN Bind
s=10.0.1.10:8076
d=192.0.2.2:6544
(5) STUN Bind
s=192.0.2.1:8076
d=192.0.2.2:6544
(6) STUN Bind
s=192.0.2.1:8076
d=192.168.1.10:6544
(7) STUN Bind
s=192.168.1.10:5556
d=192.168.1.1:1010
(8) STUN Reply
s=192.168.1.1:1010
d=192.168.1.10:5556
M=192.168.1.10:5556
(9) STUN Reply
s=192.168.1.10:6544
d=192.0.2.1:8076
M=192.168.1.10:5556
(10) STUN Reply
s=192.0.2.2:6544
d=192.0.2.1:8076
M=192.168.1.10:5556


|(11) STUN Reply
|s=192.0.2.2:6544
|d=10.0.1.10:8076
|M=192.168.1.10:5556

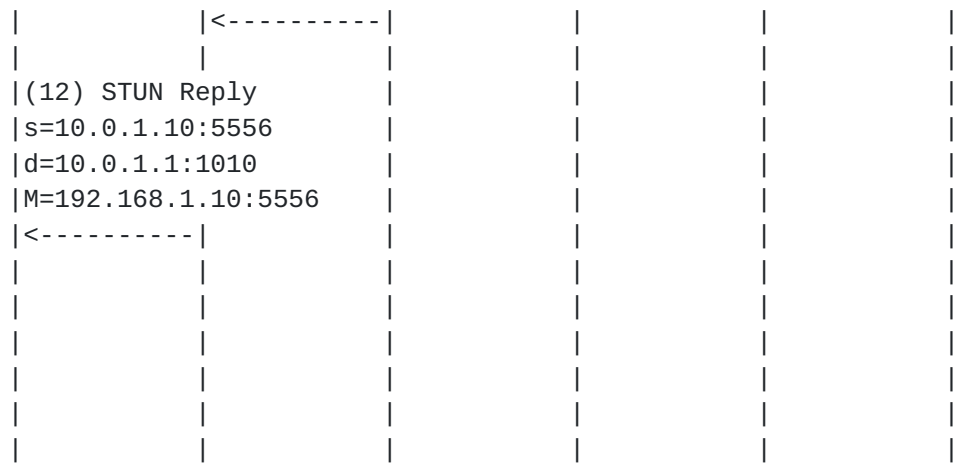
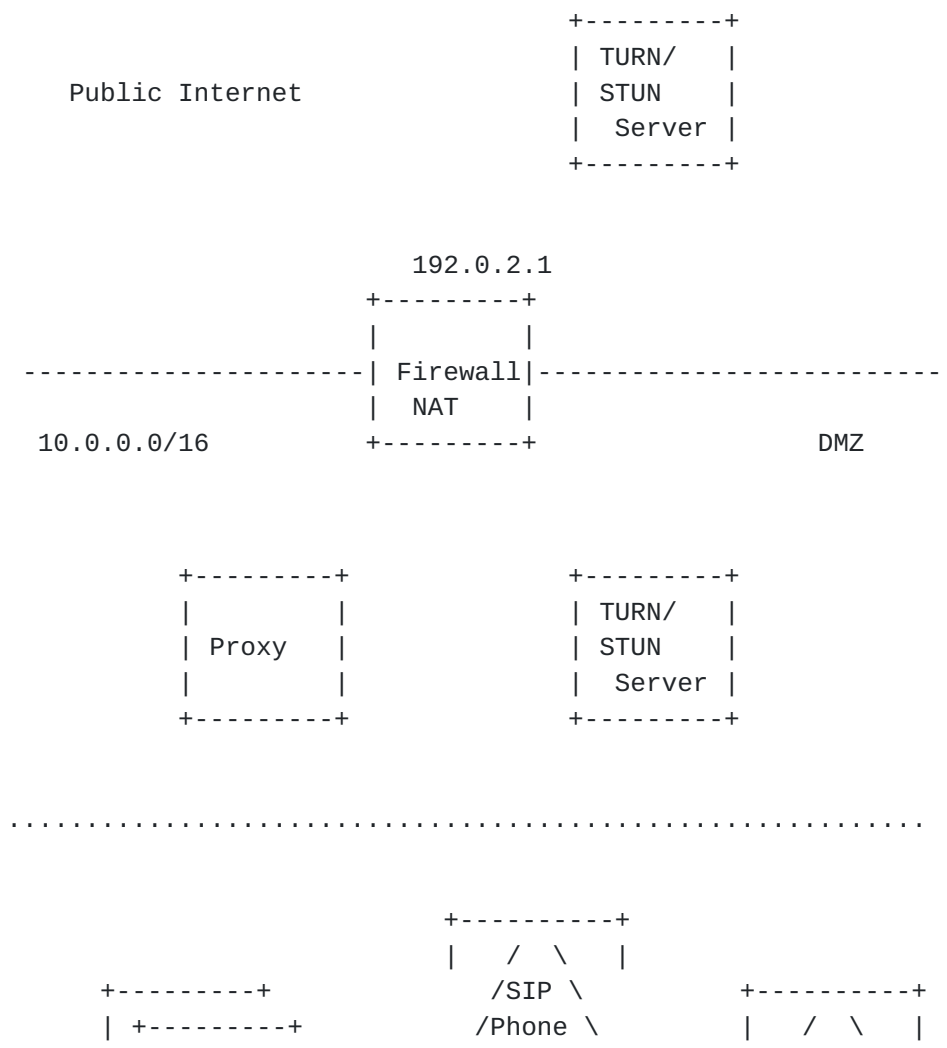


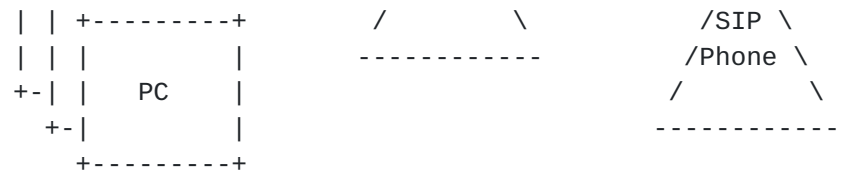
Figure 22: A's Connectivity Checks

4. Advanced Enterprise

The network of Section 3 describes a basic enterprise. It requires the enterprise to configure port forwarding on a range of external addresses, forwarding them to the internal TURN server. It also requires that ICE be deployed within the whole enterprise, since the default address won't work when talking to non-ICE clients within the enterprise.

A more complex network design can be used in enterprises that refuse to enable port forwarding/static bindings, and for which a heterogeneous internal network is expected. The design of this network is shown in Figure 23





Enterprise

Figure 23: Enterprise Configuration

In this network, there are two TURN servers. There is one internal to the firewall, and one external. Clients only contact the internal one directly. This TURN server authenticates the client, and then obtains the public binding by sending a TURN request to the external TURN server. The external TURN server returns a public address, which is forwarded to the client by the internal TURN server. The TURN query from the internal to external server creates a NAT binding in the enterprise NAT, and therefore, static bindings are no longer required. Authentication is done by the internal TURN server so that the external server does not need to contact an internal database; all database access is kept internal. The external TURN server still authenticates the TURN query, but the authentication is done using a configured username and password, configured into both the external and internal servers. For security, that username and password can be highly randomized and altered periodically - it is not used by end users, but rather by network equipment.

TODO: Add call flows.

5. Centrex

In a centrex scenario, a third party provider owns and operates the SIP and TURN/STUN servers. The enterprise merely changes their firewall configuration to allow SIP traffic out to port 5060 to the provider's SIP proxy, and to allow TURN traffic out to port 5556 and 3478, on the provider's TURN/STUN server. The corporate NAT is symmetric. The TURN/STUN server runs on 192.0.2.10. This scenario is shown in Figure 24.

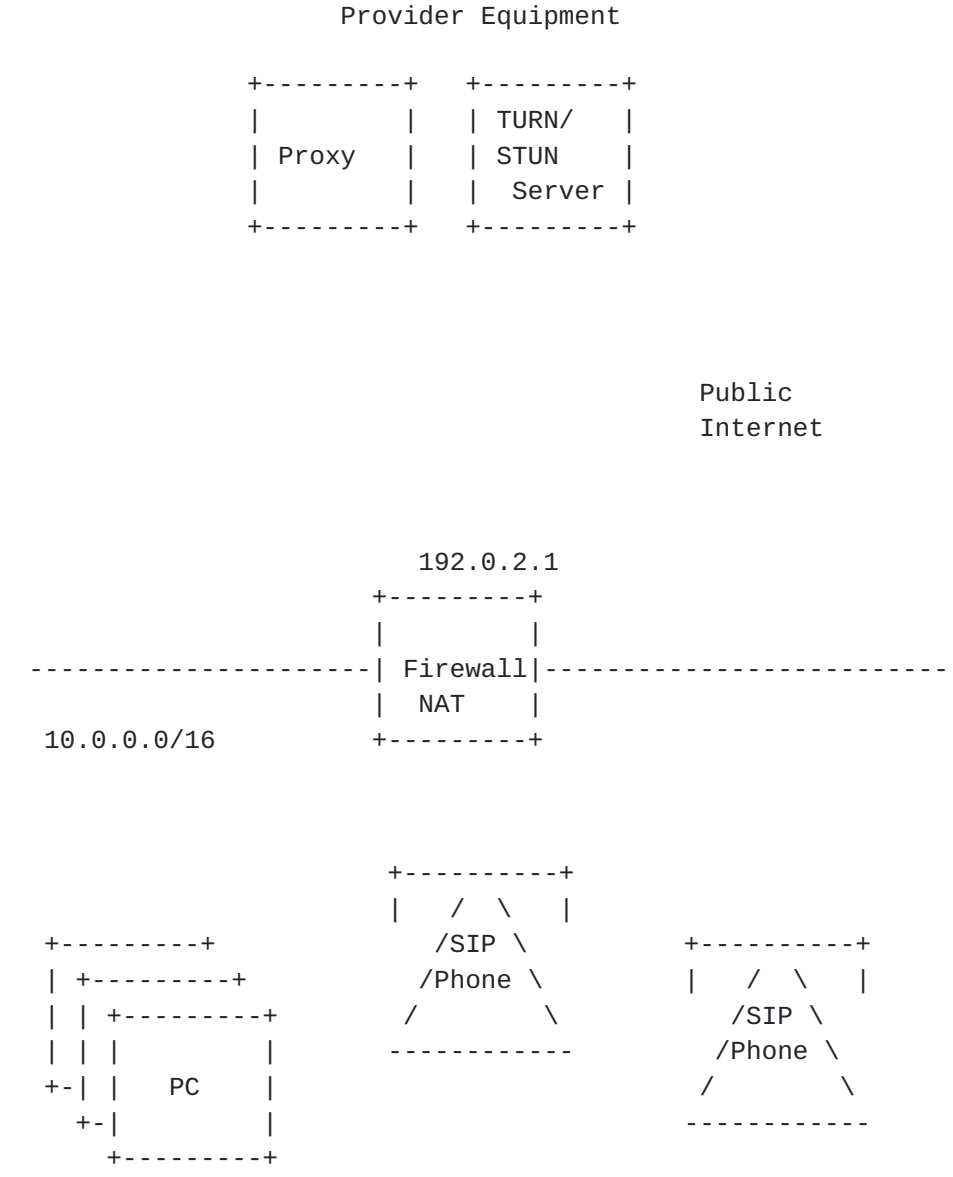
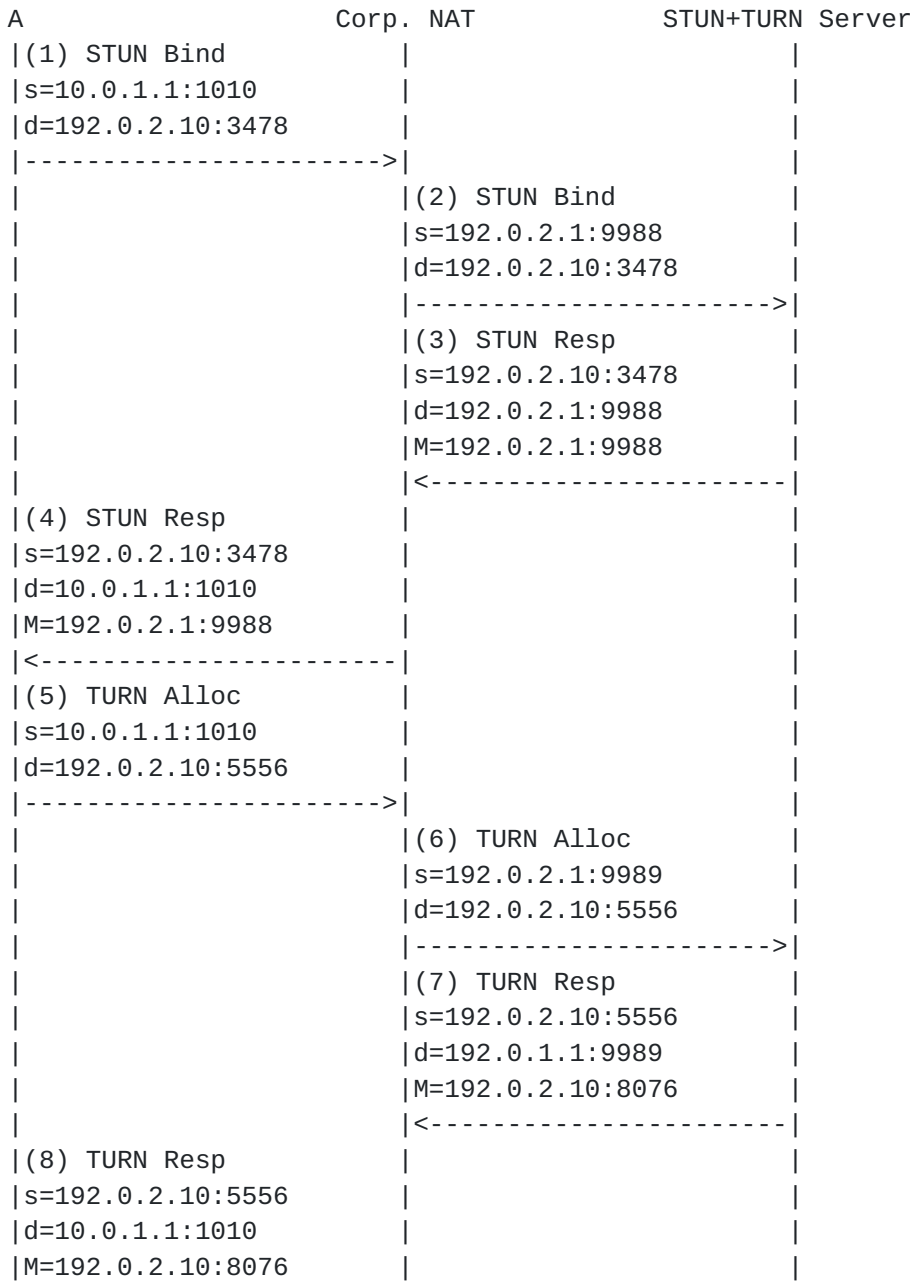


Figure 24: Centrex Configuration

5.1 Intra-Enterprise Call

In this scenario, user A calls user B. Both are within the enterprise. First, A performs its unilateral allocations. These are shown in Figure 25. These yield a STUN derived transport address and a TURN derived transport address. A sends these in the offer shown in Figure 26.



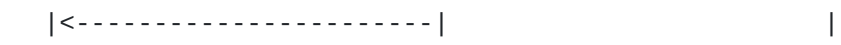
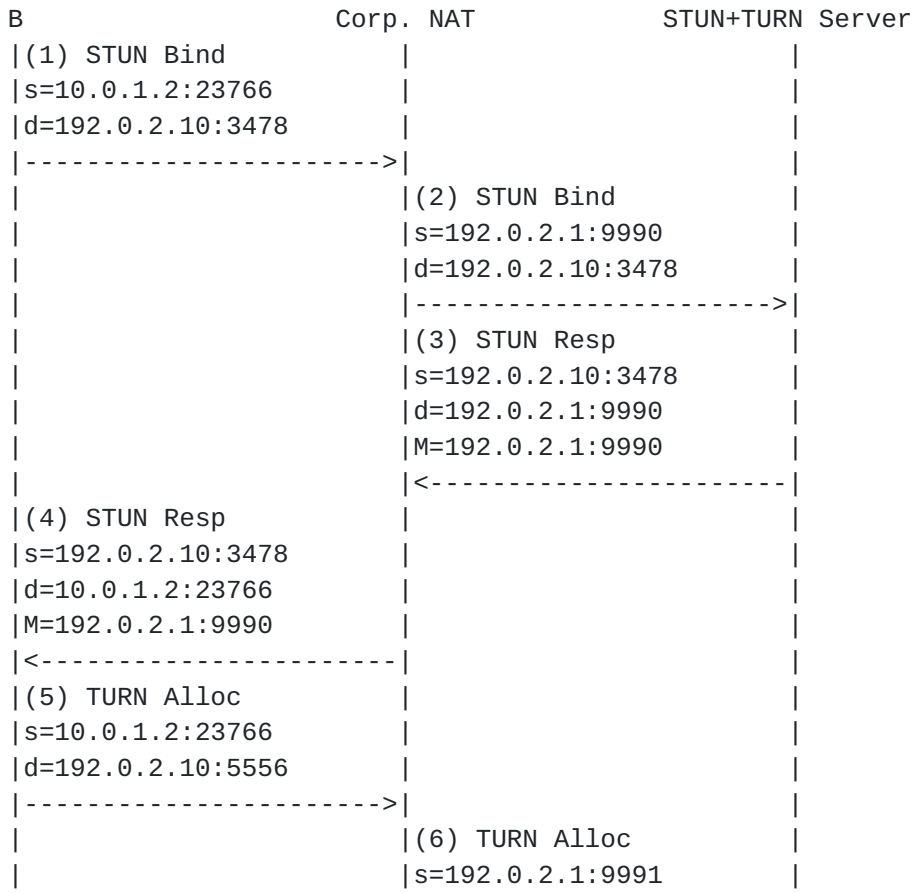


Figure 25: A's Unilateral Allocations

```
v=0
o=alice 2890844730 2890844731 IN IP4 host.example.com
s=
c=IN IP4 192.0.2.10
t=0 0
m=audio 8076 RTP/AVP 0
a=alt:1 1.0 : user 9kksj== 10.0.1.1 1010
a=alt:2 0.5 : user1 9kksk== 192.0.2.1 9988
a=alt:3 0.4 : user2 9kksl== 192.0.2.10 8076
```

Figure 26: A's Offer

This offer is received by B. B performs its unilateral allocations, shown in Figure 27. These yield a STUN derived and TURN derived transport address.



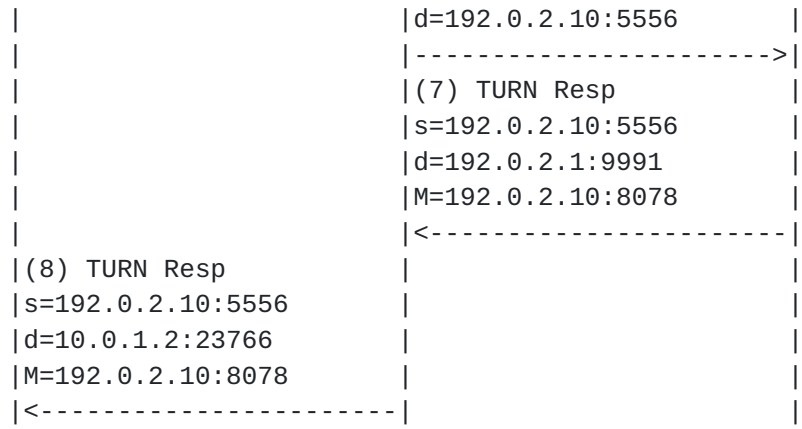
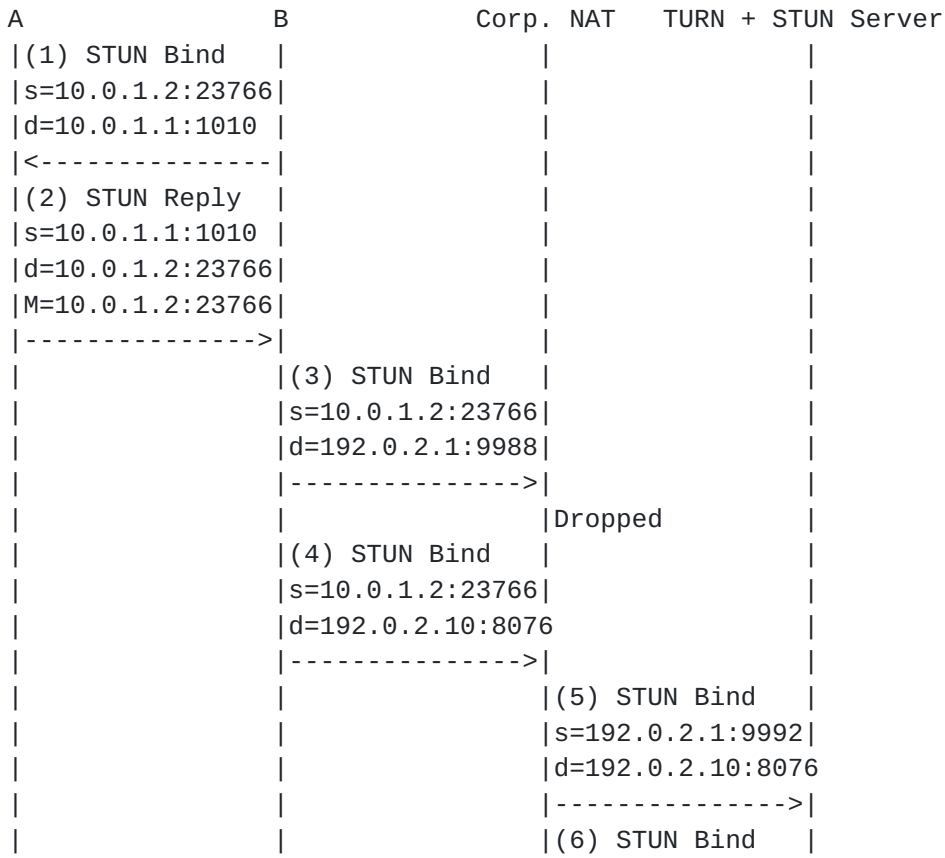


Figure 27: B's Unilateral Allocations

Now, B begins its connectivity checks, as shown in Figure 28. The first check (message 1), to A's local transport address, 10.0.1.1:1010, succeeds, since A and B are behind the same NAT. The second check, to A's STUN derived transport address, fails, since the enterprise NAT won't turn around packets. The third check, to A's TURN derived transport address, 192.0.2.10:8076, also succeeds, and yields B a new peer derived transport address, 192.0.2.10:5556.



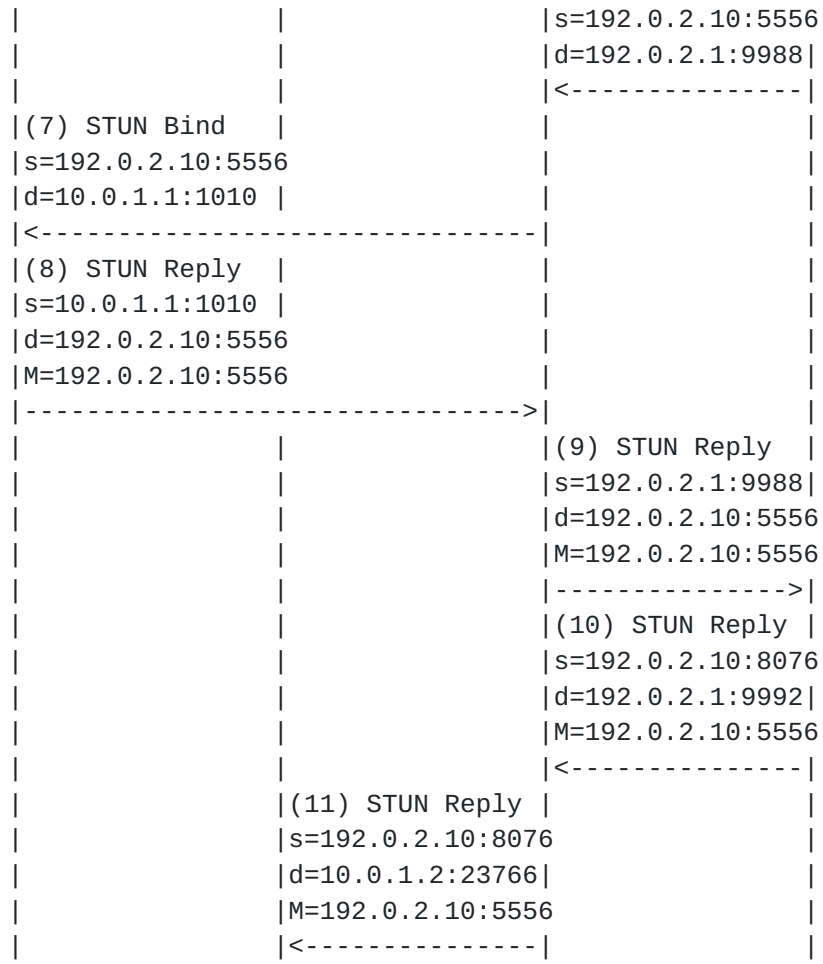


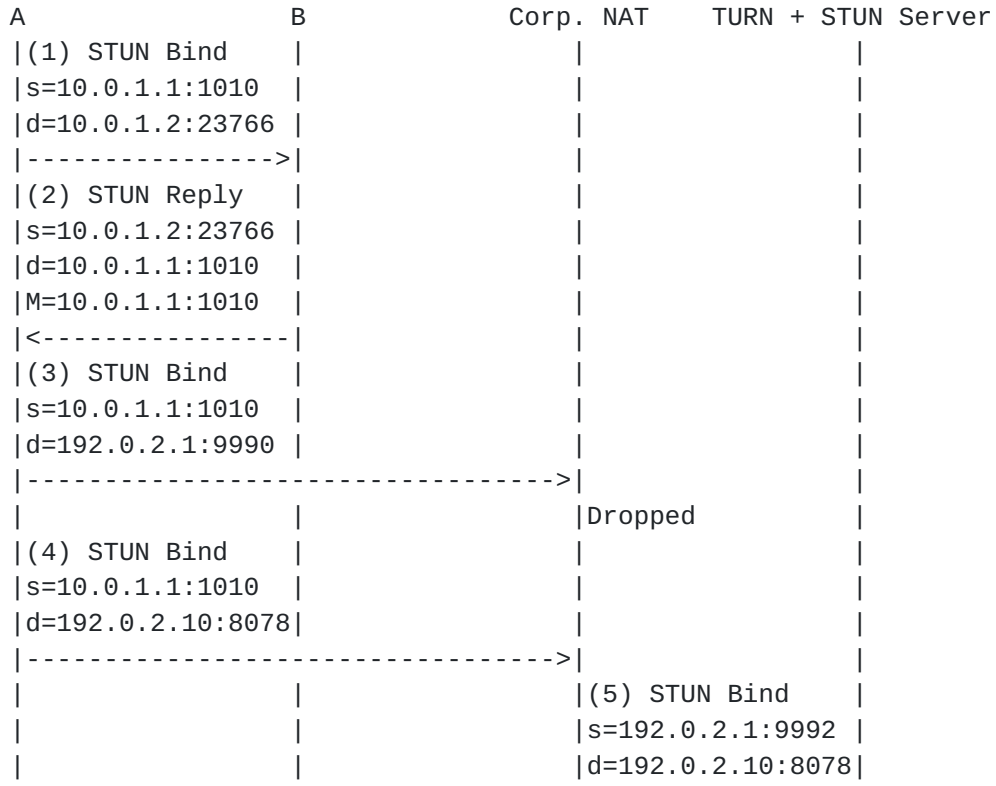
Figure 28: B's Connectivity Checks

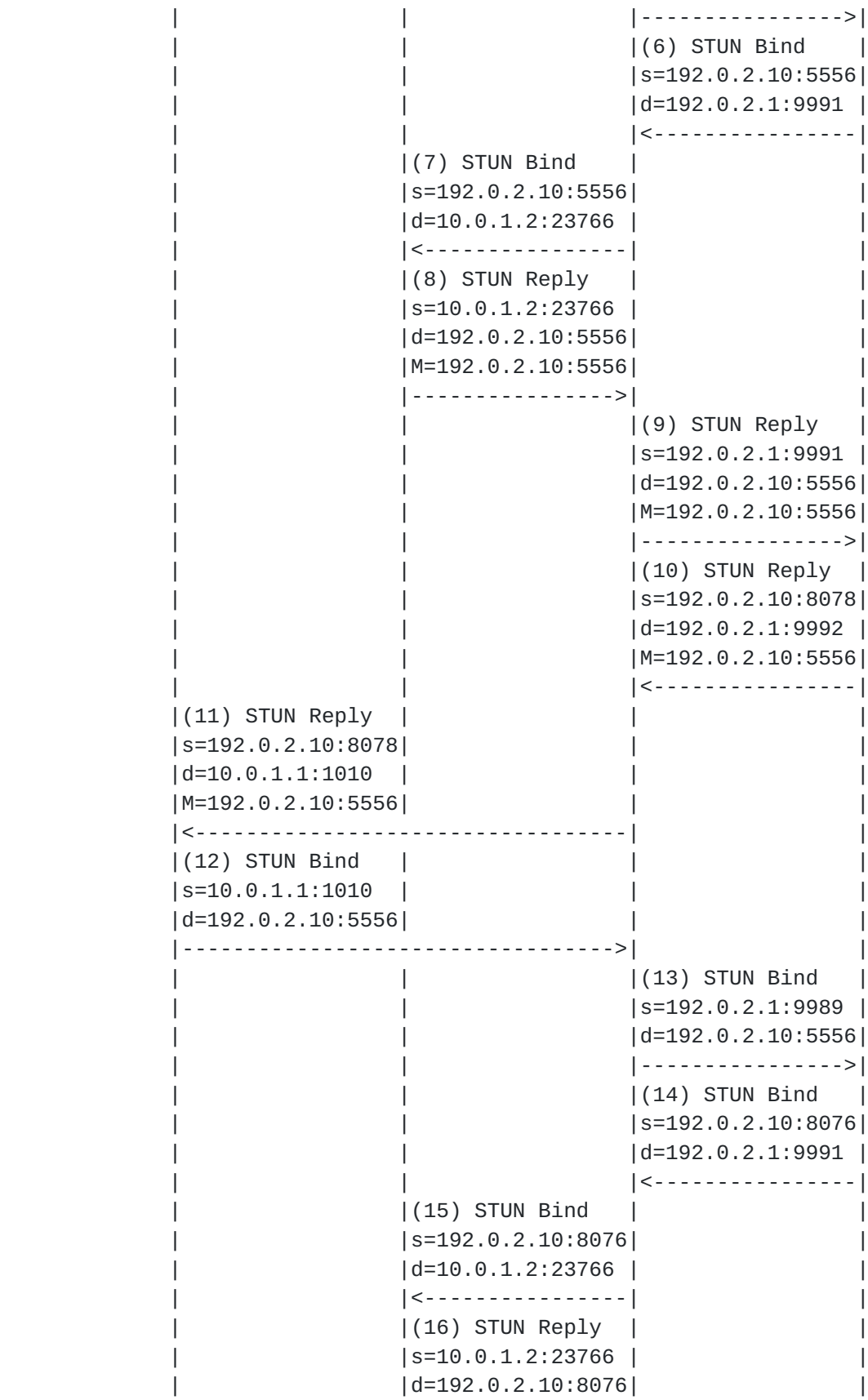
B can now send media to A directly. It also generates an answer, shown in Figure 29.

```
v=0
o=bob 2890844730 289084871 IN IP4 host2.example.com
s=
c=IN IP4 192.0.2.10
t=0 0
m=audio 8078 RTP/AVP 0
a=alt:4 1.0 : peer as88jl 10.0.1.2 23766
a=alt:5 0.8 : peer1 as88kl 192.0.2.1 9990
a=alt:6 0.4 : peer2 as88ll 192.0.2.10 8078
a=alt:7 0.4 : peer3 as88ml 192.0.2.10 5556
```

Figure 29: B's Answer

This arrives at A. A is able to send media immediately to B using the default, 192.0.2.10:8078. It also starts its connectivity checks to find a better choice. These checks are shown in Figure 30. As expected, the check for connectivity to 10.0.1.2:23766 succeeds, representing the highest priority address. The check to 192.0.2.1:9990 fails, because the NAT won't turn around internal packets. The checks to 192.0.2.10:8078 and 192.0.2.10:5556 succeed, and the former resuls in a peer-derived transport address of 192.0.2.10:5556. However, A knows that B has already connected to a higher priority address, so it doesn't bother with an additional offer/answer exchange.




(6) STUN Bind
s=192.0.2.10:5556
d=192.0.2.1:9991
(7) STUN Bind
s=192.0.2.10:5556
d=10.0.1.2:23766
(8) STUN Reply
s=10.0.1.2:23766
d=192.0.2.10:5556
M=192.0.2.10:5556
(9) STUN Reply
s=192.0.2.1:9991
d=192.0.2.10:5556
M=192.0.2.10:5556
(10) STUN Reply
s=192.0.2.10:8078
d=192.0.2.1:9992
M=192.0.2.10:5556
(11) STUN Reply
s=192.0.2.10:8078
d=10.0.1.1:1010
M=192.0.2.10:5556
(12) STUN Bind
s=10.0.1.1:1010
d=192.0.2.10:5556
(13) STUN Bind
s=192.0.2.1:9989
d=192.0.2.10:5556
(14) STUN Bind
s=192.0.2.10:8076
d=192.0.2.1:9991
(15) STUN Bind
s=192.0.2.10:8076
d=10.0.1.2:23766
(16) STUN Reply
s=10.0.1.2:23766
d=192.0.2.10:8076

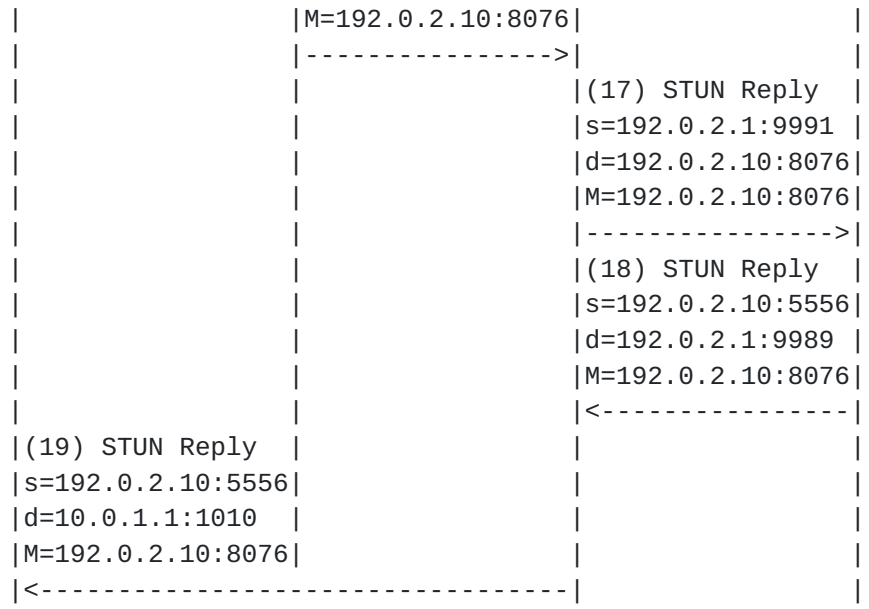


Figure 30: A's Connectivity Checks

The conclusion is that A and B communicate directly, without using the provider's relay. They can proceed to de-allocate the TURN addresses once the call is active.

6. An IPv6 Network with a pool of IPv4 addresses

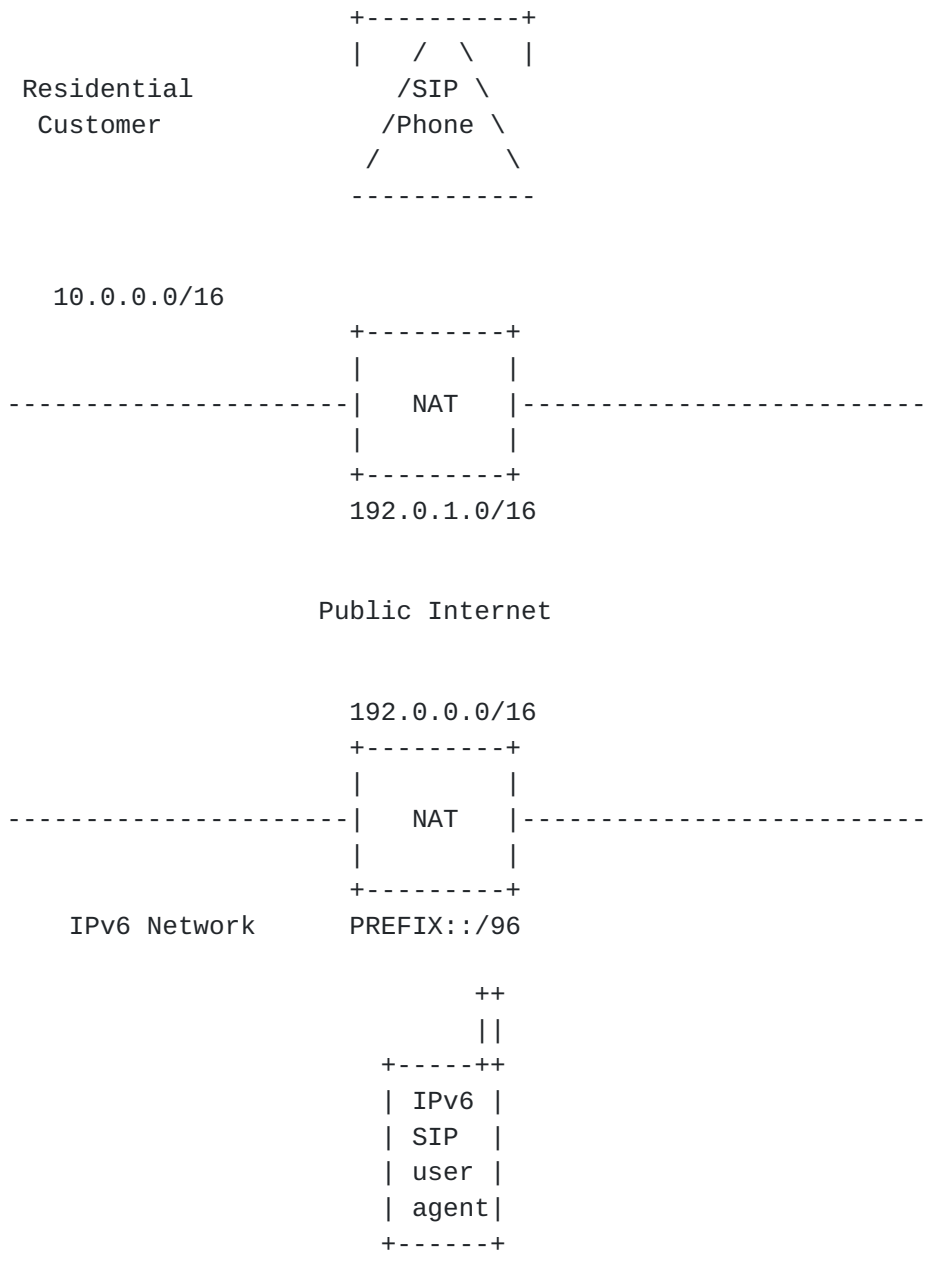


Figure 31

This example deals with a network of IPv6 SIP user agents that has a NAT with a pool of public IPv4 addresses, as shown in Figure 31. The NAT advertises the prefix PREFIX::/96 in the IPv6 network, so all

packets addresses to that PREFIX are routed to the NAT, as described in RFC 2766 [9]. The IPv6 SIP user agents of this IPv6 network need to communicate with users on the IPv4 Internet and with residential users behind a NAT (i.e., with private IPv4 addresses), even if those residential users do not have access to any STUN or TURN servers. It is assumed, though, that the residential users can run STUN servers on their ports.

For a particular session, a given IPv6 SIP user agent can obtain the services from the NAT. The NAT receives IP packets from the IPv6 SIP terminal on an IPv6 address and forwards them to the peer's IPv4 address (as seen from the NAT). It also receives packets from the peer on an IPv4 address and forwards them to the IPv6 address of the IPv6 SIP user agent.

This scenario is different from the residential user scenario described in Section 2 because the IPv6 terminal needs to communicate with the NAT to obtain a public IPv4 address to place in its offer and answers. This is because residential users would not understand IPv6 addresses in the SDP. The way the IPv6 SIP user agent obtains this IPv4 address is outside the scope of this document.

The 3G IP Multimedia Subsystem (IMS) has the characteristics just described. A solution that allows IPv6 IMS terminals to communicate with Internet users where the terminals obtain the public IPv4 address from the NAT using session policies is described in [10].

6.1 Initial Offer Generated by the IPv6 SIP User Agent

In this example, an IPv6 SIP user agent sends an offer to a residential user that is located behind a NAT. Before generating an offer, the IPv6 SIP user agent obtains a public IPv4 address from the NAT. The IPv6 SIP user agent groups both addresses (its IPv6 address and the public IPv4 address that it just obtained) using the IPV semantics [11] and places them in its offer, which is shown in Figure 32.

```
v=0
o=bob 280744730 28977631 IN IP6 host.example.com
s=
t=0 0
a=group:IPV 1 2
m=audio 6886 RTP/AVP 0
c=IN IP6 2001:056D::112E:144A:1E24
a=mid:1
m=audio 22334 RTP/AVP 0
c=IN IP4 192.0.0.1
```

```
a=mid:2
a=alt:1 1.0 : user 9kksj== 192.0.0.1 22334
```

Figure 32

When the residential user receives the offer in Figure 32, it uses STUN to obtain new addresses to place in its answer, as shown in Figure 33. The IPv6 SIP user agent responds to the residential user's STUN Bind messages with a STUN reply. This STUN reply carries a new address (192.0.1.1:2000), which the residential user places in its answer, shown in Figure 34. The answer indicates that this address has been derived from the alternative number 1 in the offer. Since the residential user does not support IPv6, it sets the port number of the media stream with the IPv6 address to zero.

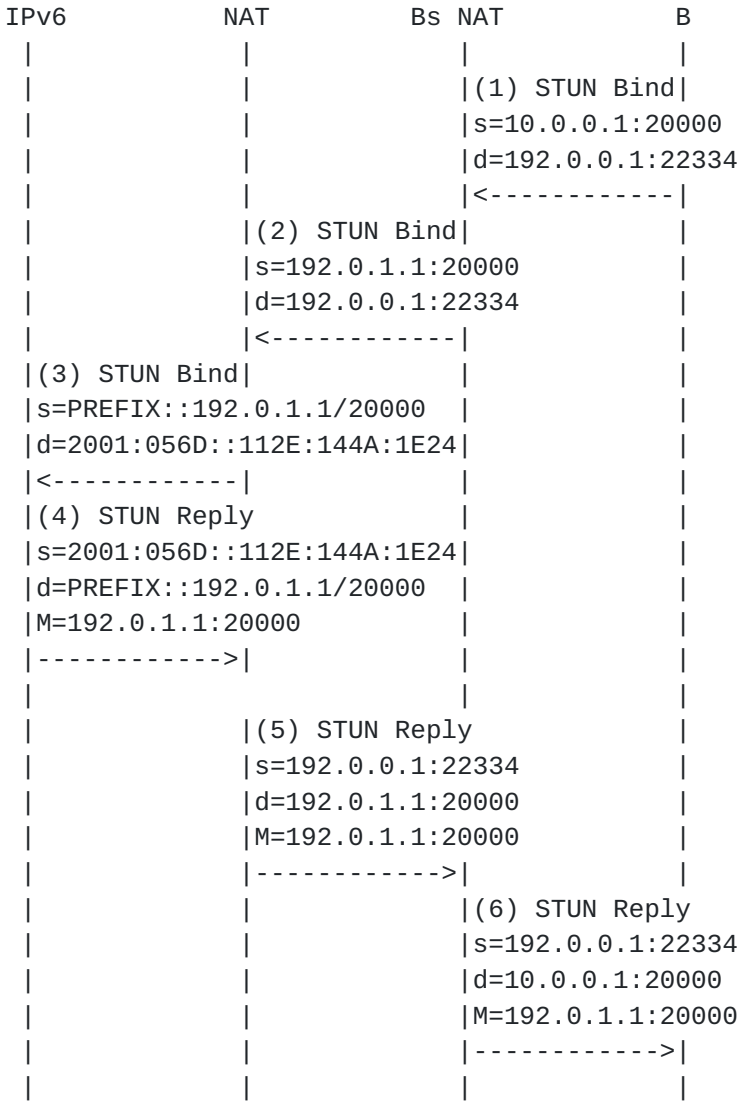


Figure 33

```
v=0
o=alice 280756730 28956631 IN IP4 host.example2.com
s=
t=0 0
m=audio 0 RTP/AVP 0
m=audio 20000 RTP/AVP 0
c=IN IP4 10.0.0.1
a=alt:2 1.0 : peer as88jl 10.0.0.1 20000
a=alt:3 0.8 1 peer as88kl 192.0.1.1 20000
```

Figure 34

When the IPv6 SIP user agent receives the answer, it uses STUN to check both addresses, 10.0.0.1:20000 and 192.0.1.1:20000. When it does so, it discovers that 10.0.0.1:20000 is unreachable and that 192.0.1.1:2000 can be used to send media to the peer.

Alternatively, the IPv6 SIP user agent could take advantage of knowing that its own IPv4 address is public and deduct which peer address to use without using STUN. If the answer contains an address which was derived from an alternative in the offer, that address will have best connectivity. If the answer does not contain any derived address, it means that the peer has a local public IPv4 address, which will be the alternative with highest priority in the answer.

6.2 Initial Offer Generated by the Residential User

In this example, a residential user that is located behind a NAT sends an offer to the IPv6 SIP user agent. The residential user places its local IPv4 address in the offer, as shown in Figure 35.

```
v=0
o=alice 280756730 28956631 IN IP4 host.example2.com
s=
t=0 0
m=audio 20000 RTP/AVP 0
c=IN IP4 10.0.0.1
a=alt:1 1.0 : peer as88jl 10.0.0.1 20000
```

Figure 35

The IPv6 SIP user agent uses STUN towards 10.0.0.1, which is unreachable. Consequently, no new addresses are discovered.

Alternatively, the IPv6 SIP user agent can skip using STUN at this point, since it knows that its NAT provides public IPv4 addresses. It does not really have any need to discover any new addresses.

The IPv6 SIP user agent places a public IPv4 address that it obtains from the NAT in its answer, as shown in Figure 36.

```
v=0
o=bob 280744730 28944631 IN IP6 host.example.com
s=
t=0 0
m=audio 22334 RTP/AVP 0
c=IN IP4 192.0.0.1
a=alt:2 1.0 : user 9kksj== 192.0.0.1 22334
```

Figure 36

When the residential user receives the answer from the IPv6 SIP user agent, it uses STUN to discover its IP address as seen by its peer (192.0.1.1:20000). The call flow is idential to the one shown in Figure 33. Then, it sends a new offer, which is shown in Figure 37.

```
v=0
o=alice 280756730 28956632 IN IP4 host.example2.com
s=
t=0 0
m=audio 20000 RTP/AVP 0
c=IN IP4 10.0.0.1
a=alt:1 1.0 : peer as88jl 10.0.0.1 20000
a=alt:3 0.8 2 peer as88kl 192.0.1.1 20000
```

Figure 37

When the IPv6 SIP user agent receives the offer in Figure 37, it uses STUN to check both addresses, 10.0.0.1:20000 and 192.0.1.1:20000. When it does so, it discovers that 10.0.0.1:20000 is unreachable and that 192.0.1.1:2000 can be used to send media to the peer.

Alternatively, the IPv6 SIP user agent could take advantage of knowing that its own IPv4 address is public and deduct which peer address to use without using STUN. If the answer contains an address which was derived from an alternative in the offer, that address will have best connectivity. If the answer does not contain any derived address, it means that the peer has a local public IPv4 address, which will be the alternative with highest priority in the answer.

At this point, the IPv6 SIP user agent sends back and answer that only differs from its previous answer (shown in Figure 36) in the version number (o= field).

7. Security Considerations

TODO.

8. IANA Considerations

There are no IANA considerations associated with this specification.

9. Acknowledgements

The authors would like to thank Douglas Otis, Karim El Malki and Francois Audet for their comments and input.

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

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

[4] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Methodology for Nettwork Address Translator (NAT) Traversal for the Session Initiation Protocol (SIP)", draft-rosenberg-sipping-ice-01 (work in progress), July 2003.

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

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

[7] Rosenberg, J., "Traversal Using Relay NAT (TURN)", draft-rosenberg-midcom-turn-04 (work in progress), February 2004.

[8] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, July 2003.

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

[10] Malki, K., "IPv6-IPv4 Translators in 3GPP Networks", draft-elmalki-v6ops-3gpp-translator-00 (work in progress), June 2003.

[11] Camarillo, G. and J. Rosenberg, "The Alternative Semantics for the Session Description Protocol Grouping Framework", draft-camarillo-mmusic-alt-02 (work in progress), October 2003.

Authors' Addresses

Jonathan Rosenberg
dynamicsoft
600 Lanidex Plaza
Parsippany, NJ 07054
US

Phone: +1 973 952-5000
EMail: jdrosen@dynamicsoft.com
URI: http://www.jdrosen.net

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.