How to Configure SIP and NAT

Sean Walberg

Abstract

Can you hear me now? Making VoIP work through a NAT gateway.

For all the technology behind Voice over IP (VoIP), you'd expect that it would work on every network, but this unfortunately isn't the case. Network Address Translation (NAT) is a common practice used in networks, and it doesn't play well with VoIP. Solving this problem requires an understanding of NAT, VoIP and your VoIP setup. This article focuses on the SIP protocol for VoIP and the Asterisk VoIP software, but the problems and solutions are applicable to most other situations.

NAT is used to hide multiple hosts behind a different set of IP addresses. As a packet leaves the gateway, the source address is rewritten to the new external address. When the return packet arrives, the gateway replaces the destination address with that of the originating host before sending the packet along its way. The gateway also can use the same external address for multiple internal hosts. The source port of the connection may be changed to ensure that the connection can be identified uniquely by the gateway, so that the return packets can have the proper address replaced.

This last scenario is also called Port Address Translation (PAT), or IP masquerading. It is what Linux and other home firewalls use to hide multiple internal hosts behind the single public IP address assigned by the carrier. The hosts must be hidden because they are using special, private, addresses that can't be routed on the Internet (such as 192.168.1.1). NAT solves the connectivity problem by giving the host a proper source address on all external connections, which allows the remote host to respond. The downside is that the inside must originate all connections in order to build the translation table entries required for NAT to work.

The problem with VoIP and NAT is that both ends of the conversation have to be able to initiate a connection to each other. Consider the simplified sequence of events that happens when PhoneA calls PhoneB using their respective SIP servers, PBXA and PBXB.

1. PBXA sends a SIP invitation to PBXB on PhoneA's behalf. In this invitation, it is PhoneA's IP address.
2. PBXB invites PhoneB to the conversation specifying PhoneA's IP address as the other end.
3. If PhoneB accepts the call, PBXB responds to PBXA with an acknowledgment that includes PhoneB's IP address.
4. PBXA tells PhoneA about PhoneB.
5. PhoneA sends audio using the Real-Time Protocol (RTP) to PhoneB.
6. PhoneB sends audio using RTP to PhoneA.

NAT can cause problems in several places. If one of the PBXes is behind a NAT gateway, the other PBX won't be able to contact it without some additional network setup. If one or more of the phones are behind a NAT gateway, the other phone will be trying to send audio to a non-routable address. This results in failed calls or missing audio.

Asterisk calls the handing off of the phone call in steps 2 and 4 above a re-invite or a native bridge. The steps above show that the phone talks to its local PBX, which in turn talks to the remote PBX. The local PBX then re-points the two sides of the call to each other, so that the local phone is talking to the remote end. Ideally, both sides will do this, and the phones are free to talk directly, leaving the SIP server out of the conversation.

The alternative to a re-invite is to have the PBX relay the voice packets between the two endpoints. We look at this in more detail later, but first, here's a more common scenario.

The simplest situation is when a SIP client is behind a NAT gateway connecting to a server on the Internet. The client creates the translation entry for the SIP traffic when it first registers. As long as there is frequent communication between the two hosts, such as one packet per minute, the channel will stay open. The only configuration needed is to have the client use its external address in all SDP packets. On clients that support it, enable STUN (Simple Traversal of UDP through NAT), so the client can determine the external address dynamically, or enter it manually. Asterisk doesn't support STUN at this time, so all NAT configuration must be done manually. The following commands in /etc/asterisk/sip.conf set up the NAT properly:

```
[general]
localnet=192.168.0.0/255.255.0.0 ; or your subnet
externip=x.x.x.x                 ; use your address

[YOURREMOTEPEER]                 ; your peer's name
nat=yes
qualify=yes                      ; Force keepalives
```

With this configuration, Asterisk uses the address defined by externip for all calls to the peers configured with nat=yes. The addition of qualify=yes causes Asterisk to test the connection frequently so that the nat translations aren't removed from the firewall. With these two commands, there always will be a communications channel between Asterisk and the peer, and Asterisk will use the outside address when sending SDP messages.

If you have multiple phones and an Asterisk server behind a NAT gateway, the solution gets more complex. Calls between the phones will work fine because NAT isn't needed. For calls between you and other systems on the Internet though, there will be problems. Unless you register to the remote side as a client (as done in the previous example), you will not be able to receive SIP messages, so you will not be able to accept calls. Second, the address information in the call setup will point to the internal address of the phone, and the one-way audio problems mentioned previously will crop up.

The easiest solution to this is to avoid NAT entirely. This may seem out of place in an article dealing with NAT, but if you have a public IP address available for your call server, use it! If your Asterisk server is connected to both the Internet and the internal network, the SIP port is reachable from both the inside and the outside, and the only problem is ensuring RTP flows properly. The PBX server need not be configured to route between the interfaces or provide masquerading; it simply needs to bridge the inbound and outbound voice calls.

As I mentioned earlier, the PBX either can stay in the voice path or get out of the way. In the latter case, the PBX tells both endpoints about each other after which the endpoints talk directly. However, Asterisk could have a call setup with both endpoints and relay the RTP packets on behalf of each endpoint. The inside host would be talking to the inside address, and the outside host would be talking to the outside address. The only configuration required to achieve this in sip.conf is to disable re-invites:

```
[general]
canreinvite=no     ; force relaying
```

This configuration works well because the Asterisk server can speak freely to the Internet to send and receive calls. It also can talk to the internal phones, and by some simple bridging, completely ignore NAT.

As it turns out, this relaying behavior also is required when the Asterisk server has only a private address. The RTP ports will have to be forwarded on the firewall too. RTP chooses random port numbers based on configured limits. Before the ports can be configured, they should be limited in range. Configuring the firewall rules is much easier if the range of ports is known beforehand.

The range of ports to be used for RTP is defined in rtp.conf. The following configuration will limit Asterisk's choice of RTP ports from 10000 to 10100:

```
[general]
rtpstart=10000 ; first port to use
rtpend=10100   ; last port to use
               ; rounded up if odd
```

Asterisk will need several RTP ports to operate properly. Only even ports are actually used, and disabling of re-invites causes two connections to be built per call. These ports and the SIP port must then be forwarded in by the firewall. The iptables syntax is:

```
iptables -t nat -A PREROUTING -i eth0 -p udp \
-m udp --dport 10000:10100 -j DNAT \
--to-destination 192.168.1.10
iptables -t nat -A PREROUTING -i eth0 -p udp \
-m udp --dport 5060 -j DNAT \
--to-destination 192.168.1.10
```

Replace eth0 with the outside interface of your firewall and 192.168.1.10 with the address of your Asterisk server. These rules tell the Linux kernel to translate the destination address of any UDP packets in the given range that are entering the outside interface. This must happen at the PREROUTING stage as opposed to the POSTROUTING stage, because the destination address is being translated. At this point, any SIP or RTP packet from the Internet will be forwarded to the internal Asterisk server for processing.

When a remote station makes a call to Asterisk, the SIP packet will be forwarded in because of the iptables rules. Asterisk will stay in the media stream because of the canreinvite=no command, and it will use the external address of the firewall in any SDP packets because of the NAT commands. Finally, the media stream will be forwarded to the Asterisk server because of the combination of iptables RTP forwarding and port ranges defined in rtp.conf.

Up to this point, the configuration has focused on getting Asterisk working behind a NAT gateway, with some extra details to make the phones relay through Asterisk. There are, of course, more general solutions.

As I said earlier, if you can avoid NAT in the first place, it's in your best interests to do so because it avoids all the problems encountered so far. The Asterisk gateway can have a very restrictive firewall policy applied to it—all that is needed is to allow UDP 5060 for SIP and whatever port range is defined in rtp.conf. In this configuration, Asterisk can contact both the internal phones and the rest of the Internet.

The most promising solution to the NAT problem is to have the firewall rewrite the SIP body as it translates the source address. The address specified for the RTP session would be replaced by the firewall itself, which also would take care of forwarding the RTP stream once it arrives. Some commercial firewalls do this. Linux iptables have shipped with ip_nat_sip and ip_conntrack_sip modules since kernel version 2.6.18. These modules are designed to take care of translating SIP, but after extensive testing, I was unable to get it working completely. I had success with inbound calls from a VoIP provider with re-invites disabled, but that was it.

IP or GRE tunnels (unencrypted) and IPSec VPNs (encrypted) are another option for getting around the need for NAT. Multiple sites are connected with a tunnel that encapsulates the internal traffic in another IP packet using the gateway's address as it leaves the outside gateway. The encapsulation is removed at the destination end. This is helpful only if you set up the tunnels beforehand. Because VPNs also are used to connect branch office data networks, this option already might be available to you. The issues of fragmentation that plague data applications aren't a problem for VoIP because small packets are used.

If SIP is not a requirement, and you're using Asterisk, consider using the IAX protocol. IAX tunnels both the control traffic and the voice traffic over a single UDP conversation that can be port-forwarded, filtered or translated easily. This method is limited to a static set of tunnels, which is sufficient if you're connecting some PBXes over the Internet or connecting to a long-distance provider.

Sometimes the above solutions aren't available to you. In that case, it might be advisable to move to a full-featured SIP proxy and use Asterisk only for voice applications, such as voice mail. SIP Express Router (SER) is a powerful SIP server that handles NAT well and is used by several high-volume services, including Free World Dialup. SER's job is only in setting up calls between endpoints, so it must rely on other applications, such as specialized media proxies, to handle RTP streams if needed.

The step beyond a SIP proxy is a Session Border Controller (SBC), which is like a VoIP firewall. The SBC can intercede in either the signaling or RTP paths to add extra features, such as signaling protocol or codec translation, all while enforcing security policies. These are almost exclusively commercial products.

It is no secret that problems will be encountered when rolling out VoIP, especially when the Internet and NAT are involved. Understanding how the protocols involved work is the first step to solving these problems. You've now seen some of the solutions, from reconfiguring the phone or PBX to port translation to additional products or even an adjustment to the architecture. With these problems out of the way, you are free to enjoy the benefits of VoIP.