

CHAPTER 8

Protocols for VoIP

The Internet is a telephone system that's gotten uppity.
—Clifford Stoll

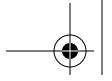
The telecommunications industry spans over 100 years, and Asterisk integrates most—if not all—of the major technologies that it has made use of over the last century. To make the most out of Asterisk, you need not be a professional in all areas, but understanding the differences between the various codecs and protocols will give you a greater appreciation and understanding of the system as a whole.

This chapter explains Voice over IP and what makes VoIP networks different from the traditional circuit-switched voice networks that were the topic of the last chapter. We will explore the need for VoIP protocols, outlining the history and potential future of each. We'll also look at security considerations and these protocols' abilities to work within topologies such as Network Address Translation (NAT). The following VoIP protocols will be discussed:

- IAX
- SIP
- H.323
- MGCP
- Skinny/SCCP
- UNISTIM

Codecs are the means by which analog voice can be converted to a digital signal and carried across the Internet. Bandwidth at any location is finite, and the number of simultaneous conversations any particular connection can carry is directly related to the type of codec implemented. In this chapter, we'll also explore the differences between the following codecs in regards to bandwidth requirements (compression level) and quality:

- G.711
- G.726



- G.723.1
- G.729A
- GSM
- iLBC
- Speex
- MP3

We will then conclude the chapter with a discussion of how voice traffic can be routed reliably, what causes echo and how to minimize it, and how Asterisk controls the authentication of inbound and outbound calls.

The Need for VoIP Protocols

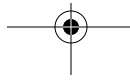
The basic premise of VoIP is the packetization* of audio streams for transport over Internet Protocol–based networks. The challenges to accomplishing this relate to the manner in which humans communicate. Not only must the signal arrive in essentially the same form that it was transmitted in, but it needs to do so in less than 300 milliseconds. If packets are lost or delayed, there will be degradation in the quality of the communications experience.

The transport protocols that collectively are called “the Internet” were not originally designed with real-time streaming of media in mind. Endpoints were expected to resolve missing packets by waiting longer for them to arrive, requesting retransmission, or, in some cases, considering the information to be gone for good and simply carrying on without it. In a typical voice conversation, these mechanisms will not serve. Our conversations do not adapt well to the loss of letters or words, nor to any appreciable delay between transmittal and receipt.

The traditional PSTN was designed specifically for the purpose of voice transmission, and it is perfectly suited to the task from a technical standpoint. From a flexibility standpoint, however, its flaws are obvious to even people with a very limited understanding of the technology. VoIP holds the promise of incorporating voice communications into all the other protocols we carry on our networks, but due to the special demands of a voice conversation, special skills are needed to design, build, and maintain these networks.

The problem with packet-based voice transmission stems from the fact that the way in which we speak is totally incompatible with the way in which IP transports data. Speaking and listening consist of the relaying of a stream of audio, whereas the

* This word hasn't quite made it into the dictionary, but it is a term that is becoming more and more common. It refers to the process of chopping a steady stream of information into discreet chunks (or *packets*), suitable for delivery independently of one another.



Internet protocols are designed to chop everything up, encapsulate the bits of information into thousands of packages, and then deliver each package in whatever way possible to the far end. Clearly, some sort of bridge was required.

VoIP Protocols

The mechanism for carrying a VoIP connection generally involves a series of signaling transactions between the endpoints (and gateways in between), culminating in two persistent media streams (one for each direction) that carry the actual conversation. There are several protocols in existence to handle this. In this section, we will discuss some of those that are important to VoIP in general and to Asterisk specifically.

IAX (The “Inter-Asterisk eXchange” Protocol)

The test of your Asterisk-ness comes when you have to pronounce the name of this protocol. Newbies say “eye-ay-ex”; those in the know say “eeks.” IAX* is an open protocol, meaning that anyone can download and develop for it, but it is not yet a standard of any kind.

In Asterisk, IAX is supported by the *chan_iax2.so* module.

History

The IAX protocol was developed by Digium for the purpose of communicating with other Asterisk servers (hence “the Inter-Asterisk eXchange protocol”). IAX is a transport protocol (much like SIP) that uses a single UDP port (4569) for both the channel signaling and Realtime Transport Protocol (RTP) streams. As discussed below, this makes it easier to firewall and more likely to work behind NAT.

IAX also has the unique ability to trunk multiple sessions into one dataflow, which can be a tremendous bandwidth advantage when sending a lot of simultaneous channels to a remote box. Trunking allows multiple data streams to be represented with a single datagram header, to lower the overhead associated with individual channels. This helps to lower latency and reduce the processing power and bandwidth required, allowing the protocol to scale much more easily with a large number of active channels between endpoints.

Future

Since IAX was optimized for voice, it has received some criticism for not better supporting video—but in fact, IAX holds the potential to carry pretty much any media

* Officially, the current version is IAX2, but all support for IAX1 has been dropped, so whether you say “IAX” or “IAX2,” it is expected that you are talking about the Version 2.

stream desired. Because it is an open protocol, future media types are certain to be incorporated as the community desires them.

Security considerations

IAX includes the ability to authenticate in three ways: plain text, MD5 hashing, and RSA key exchange. This, of course, does nothing to encrypt the media path or headers between endpoints. Many solutions include using a Virtual Private Network (VPN) appliance or software to encrypt the stream in another layer of technology, which requires the endpoints to pre-establish a method of having these tunnels configured and operational. In the future, IAX may be able to encrypt the streams between endpoints with the use of an exchanged RSA key, or dynamic key exchange at call setup, allowing the use of automatic key rollover. This would be very attractive for creating a secure link with an institution such as your bank. The various law enforcement agencies, however, are going to want some level of access to such connections.

IAX and NAT

The IAX2 protocol was deliberately designed to work from behind devices performing NAT. The use of a single UDP port for both signaling and transmission of media also keeps the number of holes required in your firewall to a minimum. These considerations have helped make IAX one of the easiest protocols (if not the easiest) to implement in secure networks.

SIP




The Session Initiation Protocol (SIP) has taken the world of VoIP by storm. Originally considered little more than an interesting idea, SIP now seems poised to dethrone the mighty H.323 as the VoIP protocol of choice—certainly at the endpoints of the network. The premise of SIP is that each end of a connection is a peer, and the protocol negotiates capabilities between them. What makes SIP compelling is that it is a relatively simple protocol, with a syntax similar to that of other familiar protocols such as HTTP and SMTP.

SIP is supported in Asterisk with the *chan_sip.so* module.

History

SIP was originally submitted to the Internet Engineering Task Force (IETF) in February of 1996 as “draft-ietf-mmusic-sip-00.” The initial draft looked nothing like the SIP we know today and contained only a single request type: a call setup request. In March of 1999, after 11 revisions, SIP RFC 2543 was born.

At first, SIP was all but ignored, as H.323 was considered the protocol of choice for VoIP transport negotiation. However, as the buzz grew, SIP began to gain popularity,




and while there may be a lot of different factors that accelerated its growth, we'd like to think that a large part of its success is due to its freely available specification.

Future


SIP has earned its place as the protocol that justified VoIP. All new user and enterprise products are expected to support SIP, and any existing products will now be a tough sell unless a migration path to SIP is offered. SIP is widely expected to deliver far more than VoIP capabilities, including the ability to transmit video, music, and any type of real-time multimedia. SIP is poised to deliver the majority of new applications over the next few years.

Security considerations

SIP uses a challenge/response system to authenticate users. An initial INVITE is sent to the proxy with which the end device wishes to communicate. The proxy then sends back a 407 Proxy Authorization Request message, which contains a random set of characters referred to as a “nonce.” This nonce is used along with the password to generate an MD5 hash, which is then sent back in the subsequent INVITE. Assuming the MD5 hash matches the one that the proxy generated, the client is then authenticated.



Denial of Service (DoS) attacks are probably the most common type of attack on VoIP communications. A DoS attack can occur when a large number of invalid INVITE requests are sent to a proxy server in an attempt to overwhelm the system. These attacks are relatively simple to implement, and their effects on the users of the system are immediate. SIP has several methods of minimizing the effects of DoS attacks, but ultimately they are impossible to prevent.



SIP implements a scheme to guarantee that a secure, encrypted transport mechanism (namely Transport Layer Security, or TLS) is used to establish communication between the caller and the domain of the callee. Beyond that, the request is sent securely to the end device, based upon the local security policies of the network. Note that the encryption of the media (that is, the RTP stream) is beyond the scope of SIP itself and must be dealt with separately.

More information regarding SIP security considerations, including registration hijacking, server impersonation, and session teardown, can be found in Section 26 of SIP RFC 3261.

SIP and NAT

Probably the biggest technical hurdle SIP has to conquer is the challenge of carrying out transactions across a NAT layer. Because SIP encapsulates addressing information in its data frames, and NAT happens at a lower network layer, the addressing information is not modified, and thus the media streams will not have the correct addressing

information needed to complete the connection when NAT is in place. In addition to this, the firewalls normally integrated with NAT will not consider the incoming media stream to be part of the SIP transaction, and will block the connection.

H.323

This International Telecommunication Union (ITU) protocol was originally designed to provide an IP transport mechanism for video-conferencing. It has become the standard in IP-based video-conferencing equipment, and it briefly enjoyed fame as a VoIP protocol as well. While there is much heated debate over whether SIP or H.323 (or IAX) will dominate the VoIP protocol world, in Asterisk, H.323 has largely been deprecated in favour of IAX and SIP. H.323 has not enjoyed much success among users and enterprises, although it is still the most widely used VoIP protocol among carriers.

The two versions of H.323 supported in Asterisk are handled by the modules *chan_h323.so* (supplied with Asterisk) and *chan_oh323.so* (available as a free add-on).



You have probably used H.323 without even knowing it—Microsoft's NetMeeting client is arguably the most widely deployed H.323 client.

History

H.323 was developed by the ITU in May of 1996 as a means to transmit voice, video, data, and fax communications across an IP-based network while maintaining connectivity with the PSTN. Since that time, H.323 has gone through several versions and annexes (which add functionality to the protocol), allowing it to operate in pure VoIP networks and more widely distributed networks.

Future

The future of H.323 is a subject of hot debate. If the media is any measure, it doesn't look good for H.323; it hardly ever gets mentioned (certainly not with the regularity of SIP). H.323 is commonly regarded as technically superior to SIP, but, as with so many other technologies, that ultimately might not matter. One of the factors that makes H.323 unpopular is its complexity—although many argue that the once-simple SIP is starting to suffer from the same problem.

H.323 still carries by far the majority of worldwide carrier VoIP traffic, but as people become less and less dependent on traditional carriers for their telecom needs, the future of H.323 becomes more difficult to predict with any certainty. While H.323 may not be the protocol of choice for new implementations, we can certainly expect to have to deal with H.323 interoperability issues for some time to come.

Security considerations

H.323 is a relatively secure protocol and does not require many security considerations beyond those that are common to any network communicating with the Internet. Since H.323 uses the RTP protocol for media communications, it does not natively support encrypted media paths. The use of a VPN or other encrypted tunnel between endpoints is the most common way of securely encapsulating communications. Of course, this has the disadvantage of requiring the establishment of these secure tunnels between endpoints, which may not always be convenient (or even possible). As VoIP becomes used more often to communicate with financial institutions such as banks, we're likely to require extensions to the most commonly used VoIP protocols to natively support strong encryption methods.

H.323 and NAT

The H.323 standard uses the Internet Engineering Task Force (IETF) RTP protocol to transport media between endpoints. Because of this, H.323 has the same issues as SIP when dealing with network topologies involving NAT. The easiest method is to simply forward the appropriate ports through your NAT device to the internal client.

To receive calls, you will always need to forward TCP port 1720 to the client. In addition, you will need to forward the UDP ports for the RTP media and RTCP control streams (see the manual for your device for the port range it requires). Older clients, such as MS Netmeeting, will also require TCP ports forwarded for H.245 tunneling (again, see your client's manual for the port number range).

If you have a number of clients behind the NAT device, you will need to use a *gatekeeper* running in proxy mode. The gatekeeper will require an interface attached to the private IP subnet and the public Internet. Your H.323 client on the private IP subnet will then register to the gatekeeper, which will proxy calls on the clients' behalf. Note that any external clients that wish to call you will also be required to register with the proxy server.

At this time, Asterisk can't act as an H.323 gatekeeper. You'll have to use a separate application, such as the open source OpenH323 Gatekeeper (<http://www.gnugk.org>).

MGCP

The Media Gateway Control Protocol (MGCP) also comes to us from the IETF. While MGCP deployment is more widespread than one might think, it is quickly losing ground to protocols such as SIP and IAX. Still, Asterisk loves protocols, so naturally it has rudimentary support for it.

MGCP is defined in RFC 3435.* It was designed to make the end devices (such as phones) as simple as possible, and have all the call logic and processing handled by media gateways and call agents. Unlike SIP, MGCP uses a centralized model. MGCP phones cannot directly call other MGCP phones; they must always go through some type of controller.

Asterisk supports MGCP through the *chan_mgcp.so* module, and the endpoints are defined in the configuration file *mgcp.conf*. Since Asterisk provides only basic call agent services, it cannot emulate an MGCP phone (to register to another MGCP controller as a user agent, for example).

If you have some MGCP phones lying around, you will be able to use them with Asterisk. If you are planning to put MGCP phones into production on an Asterisk system, keep in mind that the community has moved on to more popular protocols, and you will therefore need to budget your software support needs accordingly. If possible (for example, with Cisco phones), you should upgrade MGCP phones to SIP.

Proprietary Protocols

Finally, let's take a look at two proprietary protocols that are supported in Asterisk.

Skinny/SCCP

The Skinny Client Control Protocol (SCCP) is proprietary to Cisco VoIP equipment. It is the default protocol for endpoints on a Cisco Call Manager PBX. Skinny is supported in Asterisk, but if you are connecting Cisco phones to Asterisk, it is generally recommended that you obtain SIP images for any phones that support it and connect via SIP instead.

UNISTIM

Support for Nortel's proprietary VoIP protocol, UNISTIM, has recently been added to Asterisk. This remarkable milestone means that Asterisk is the first PBX in history to natively support proprietary IP terminals from the two biggest players in VoIP, Nortel and Cisco.

Codecs

Codecs are generally understood to be various mathematical models used to digitally encode (and compress) analog audio information. Many of these models take into account the human brain's ability to form an impression from incomplete information. We've all seen optical illusions; likewise, voice-compression algorithms take

* RFC 3435 obsoletes RFC 2705.

advantage of our tendency to interpret what we *believe* we should hear, rather than what we *actually* hear.* The purpose of the various encoding algorithms is to strike a balance between efficiency and quality.†

Originally, the term CODEC referred to a Coder/DECoder: a device that converts between analog and digital. Now, the term seems to relate more to COmpression/DECompression.

Before we dig into the individual codecs, take a look at Table 8-1—it’s a quick reference that you may want to refer back to.

Table 8-1. Codec quick reference

Codec	Data bitrate (kbps)	Licence required?
G.711	64 kbps	No
G.726	16, 24, or 32 kbps	No
G.723.1	5.3 or 6.3 kbps	Yes (no for passthrough)
G.729A	8 kbps	Yes (no for passthrough)
GSM	13 kbps	No
iLBC	13.3 kbps (30-ms frames) or 15.2 kbps (20-ms frames)	No
Speex	Variable (between 2.15 and 22.4 kbps)	No

G.711

G.711 is the fundamental codec of the PSTN. In fact, if someone refers to PCM (discussed in the previous chapter) with respect to a telephone network, you are allowed to think of G.711. Two companding methods are used: μ -law in North America and A-law in the rest of the world. Either one delivers an 8-bit word transmitted 8,000 times per second. If you do the math, you will see that this requires 64,000 bits to be transmitted per second.

Many people will tell you that G.711 is an uncompressed codec. This is not exactly true, as companding is considered a form of compression. What is true is that G.711 is the base codec from which all of the others are derived.

* “Aoccdmrig to rsreach at an Elingsh uinervtisy, it deosn’t mtttaer in waht oredr the ltteers in a wrod are, the olny iprmoentn tihng is taht frist and lsat lttteres are in the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit a porbelm. Tihs is bcuseae we do not raed ervey lteter by istlef, but the wrod as a wlohe.” (The source of this quote is unknown—see http://www.bisso.com/ujg_archives/000228.html.) Tihs is ture with snoud, too.

† On an audio CD, quality is far more important than bandwidth, so the audio is quantized at 16 bits (times 2, as it’s stereo), with a sampling rate of 44,100 Hz. Considering that the CD was invented in the late 1970s, this was quite impressive stuff. The telephone network does not require this level of quality (and needs to optimize bandwidth), so telephone signals are encoded using 8 bits, at a sampling frequency of 8,000 Hz.

G.726

This codec has been around for some time (it used to be G.721, which is now obsolete), and it is one of the original compressed codecs. It is also known as Adaptive Differential Pulse-Code Modulation (ADPCM), and it can run at several bitrates. The most common rates are 16 kbps, 24 kbps, and 32 kbps. As of this writing, Asterisk currently supports only the ADPCM-32 rate, which is far and away the most popular rate for this codec.

G.726 offers quality nearly identical to G.711, but it uses only half the bandwidth. This is possible because rather than sending the result of the quantization measurement, it sends only enough information to describe the difference between the current sample and the previous one. G.726 fell from favor in the 1990s due to its inability to carry modem and fax signals, but because of its bandwidth/CPU performance ratio it is now making a comeback. G.726 is especially attractive because it does not require a lot of computational work from the system.

G.723.1

Not to be confused with G.723 (which is another obsolete version of ADPCM), this codec is designed for low-bitrate speech. It has two data bitrate settings: 5.3 kbps and 6.3 kbps. G.723.1 is one of the codecs required for compliance with the H.323 protocol (although other codecs may be employed by H.323). It is currently encumbered by patents and thus requires licensing if used in commercial applications. What this means is that while you can switch two G.723.1 calls through your Asterisk system, you are not allowed to decode them without a license.

G.729A

Considering how little bandwidth it uses, G.729A delivers impressive sound quality. It does this through the use of Conjugate-Structure Algebraic-Code-Excited Linear Prediction (CS-ACELP).^{*} Because of patents, you can't use G.729A without paying a licensing fee; however, it is extremely popular and is thus well supported on many different phones and systems.

To achieve its impressive compression ratio, this codec requires an equally impressive amount of effort from the CPU. In an Asterisk system, the use of heavily compressed codecs will quickly bog down the CPU.

G.729A uses 8 kbps of bandwidth.

^{*} CELP is a popular method of compressing speech. By mathematically modeling the various ways humans make sounds, a codebook of sounds can be built. Rather than sending an actual sampled sound, a code corresponding to the sound is then sent. (Of course, there is much more to it than that.) Jason Woodward's Speech Coding page (http://www-mobile.ecs.soton.ac.uk/speech_codecs/) is a source of helpful information for the non-mathematically inclined. This is fairly heavy stuff, though, so wear your thinking cap.

GSM

GSM is the darling codec of Asterisk. This codec does not come encumbered with a licensing requirement the way that G.723.1 and G.729A do, and it offers outstanding performance with respect to the demand it places on the CPU. The sound quality is generally considered to be of a lesser grade than that produced by G.729A, but as much of this comes down to personal opinion, be sure to try it out.

GSM operates at 13 kbps.

iLBC

The Internet Low Bitrate Codec (iLBC) provides an attractive mix of low bandwidth usage and quality, and it is especially well suited to sustaining reasonable quality on lossy network links.

Naturally, Asterisk supports it (and support elsewhere is growing), but it is not as popular as the ITU codecs and thus may not be compatible with common IP telephones and commercial VoIP systems. IETF RFCs 3951 and 3952 have been published in support of iLBC, and iLBC is on the IETF standards track.

Because iLBC uses complex algorithms to achieve its high levels of compression, it has a fairly high CPU cost in Asterisk.

While you are allowed to use iLBC without paying royalty fees, the holder of the iLBC patent, Global IP Sound (GIPS), wants to know whenever you use it in a commercial application. The way you do that is by downloading and printing a copy of the iLBC license, signing it, and returning it to them. If you want to read about iLBC and its license, you can do so at <http://www.ilbcfreeware.org>.

iLBC operates at 13.3 kbps (30-ms frames) and 15.2 kbps (20-ms frames).

Speex

Speex is a Variable Bitrate (VBR) codec, which means that it is able to dynamically modify its bitrate to respond to changing network conditions. It is offered in both narrowband and wideband versions, depending on whether you want telephone quality or better.

Speex is a totally free codec, licensed under the Xiph.org variant of the BSD license.

An Internet draft for Speex is available, and more information about Speex can be found at its home page (<http://www.speex.org>).

Speex can operate at anywhere from 2.15 to 22.4 kbps, due to its variable bitrate

MP3

Sure thing, MP3 is a codec. Specifically, it's the Moving Picture Experts Group Audio Layer 3 Encoding Standard.* With a name like that, it's no wonder we call it MP3! In Asterisk, the MP3 codec is typically used for Music on Hold (MoH). MP3 is not a telephony codec, as it is optimized for music, not voice; nevertheless, it's very popular with VoIP telephony systems as a method of delivering Music on Hold.



Be aware that music cannot usually be broadcast without a license. Many people assume that there is no legal problem with connecting a radio station or CD as a Music on Hold source, but this is very rarely true.

Quality of Service

Quality of Service, or QoS as it's more popularly termed, refers to the challenge of delivering a time-sensitive stream of data across a network that was designed to deliver data in an ad hoc, best-effort sort of way. Although there is no hard rule, it is generally accepted that if you can deliver the sound produced by the speaker to the listener's ear within 300 milliseconds, a normal flow of conversation is possible. When delay exceeds 500 milliseconds, it becomes difficult to avoid interrupting each other. Beyond one second, normal conversation becomes extremely awkward.

In addition to getting it there on time, it is also essential to ensure that the transmitted information arrives intact. Too many lost packets will prevent the far end from completely reproducing the sampled audio, and gaps in the data will be heard as static or, in severe cases, entire missed words or sentences.

TCP, UDP, and SCTP

If you're going to send data on an IP-based network, it will be transported using one of the three transport protocols discussed here.

Transmission Control Protocol

The Transmission Control Protocol (TCP) is almost never used for VoIP, for while it does have mechanisms in place to ensure delivery, it is not inherently in any hurry to do so. Unless you have an extremely low-latency interconnection between the two endpoints, TCP is going to tend to cause more problems than it solves.

The purpose of TCP is to guarantee the delivery of packets. In order to do this, several mechanisms are implemented, such as packet numbering (for reconstructing

* If you want to learn all about MPEG audio, do a web search for Davis Pan's paper entitled "A Tutorial on MPEG/Audio Compression."

blocks of data), delivery acknowledgment, and re-requesting lost packets. In the world of VoIP, getting the packets to the endpoint quickly is paramount—but 20 years of cellular telephony has trained us to tolerate a few lost packets.*

TCP's high processing overhead, state management, and acknowledgment of arrival work well for transmitting large amounts of data, but it simply isn't efficient enough for real-time media communications.

User Datagram Protocol

Unlike TCP, the User Datagram Protocol (UDP) does not offer any sort of delivery guarantee. Packets are placed on the wire as quickly as possible and released into the world to find their way to their final destinations, with no word back as to whether they get there or not. Since UDP itself does not offer any kind of guarantee that the data will arrive,[†] it achieves its efficiency by spending very little effort on what it is transporting.



TCP is a more “socially responsible” protocol, because the bandwidth is more evenly distributed to clients connecting to a server. As the percentage of UDP traffic increases, it is possible that a network could become overwhelmed.

Stream Control Transmission Protocol

Approved by the IETF as a proposed standard in RFC 2960, SCTP is a relatively new transport protocol. From the ground up, it was designed to address the shortcomings of both TCP and UDP, especially as related to the types of services that used to be delivered over circuit-switched telephony networks.

Some of the goals of SCTP were:

- Better congestion-avoidance techniques (specifically, avoiding Denial of Service attacks)
- Strict sequencing of data delivery
- Lower latency for improved real-time transmissions

By overcoming the major shortcomings of TCP and UDP, the SCTP developers hoped to create a robust protocol for the transmission of SS7 and other types of PSTN signaling over an IP-based network.

* The order of arrival is important in voice communication, because the audio will be processed and sent to the caller ASAP. However, with a jitter buffer the order of arrival isn't as important, as it provides a small window of time in which the packets can be reordered before being passed on to the caller.

† Keep in mind that the upper-layer protocols or applications can implement their own packet acknowledgment systems.

Differentiated Service

Differentiated service, or DiffServ, is not so much a QoS mechanism as a method by which traffic can be flagged and given specific treatment. Obviously, DiffServ can help to provide QoS by allowing certain types of packets to take precedence over others. While this will certainly increase the chance of a VoIP packet passing quickly through each link, it does not guarantee anything.

Guaranteed Service

The ultimate guarantee of QoS is provided by the PSTN. For each conversation, a 64-kbps channel is completely dedicated to the call—the bandwidth is guaranteed. Similarly, protocols that offer guaranteed service can ensure that a required amount of bandwidth is dedicated to the connection being served. As with any packetized networking technology, these mechanisms generally operate best when traffic is below maximum levels. When a connection approaches its limits, it is next to impossible to eliminate degradation.

MPLS

Multiprotocol Label Switching (MPLS) is a method for engineering network traffic patterns independent of layer-3 routing tables. The protocol works by assigning short labels (MPLS frames) to network packets, which routers then use to forward the packets to the MPLS egress router, and ultimately to their final destinations. Traditionally, routers make an independent forwarding decision based on an IP table lookup at each hop in the network. In an MPLS network, this lookup is performed only once, when the packet enters the MPLS cloud at the ingress router. The packet is then assigned to a stream, referred to as a Label Switched Path (LSP), and identified by a label. The label is used as a lookup index in the MPLS forwarding table, and the packet traverses the LSP independent of layer-3 routing decisions. This allows the administrators of large networks to fine-tune routing decisions and to make the best use of network resources. Additionally, information can be associated with a label to prioritize packet forwarding.

RSVP

MPLS contains no method to dynamically establish LSPs, but you can use the Reservation protocol (RSVP) with MPLS. RSVP is a signaling protocol used to simplify the establishment of LSPs and to report problems to the MPLS ingress router. The advantage of using RSVP in conjunction with MPLS is the reduction in administrative overhead. If you don't use RSVP with MPLS, you'll have to go to every single router and configure the labels and each path manually. Using RSVP makes the network more dynamic by distributing control of labels to the routers. This enables the network to become more responsive to changing conditions, because it can be set up to change the paths based on certain conditions, such as a certain path going down

(perhaps due to a faulty router). The configuration within the router will then be able to use RSVP to distribute new labels to the routers in the MPLS network, with no (or minimal) human intervention.

Best Effort

The simplest, least expensive approach to QoS is not to provide it at all—the “best effort” method. While this might sound like a bad idea, it can in fact work very well. Any VoIP call that traverses the public Internet is almost certain to be best effort, as QoS mechanisms are not yet common in this environment.

Echo

You may not realize it, but echo has been a problem in the PSTN for as long as there have been telephones. You probably haven’t often experienced it, because the telecom industry has spent large sums of money designing expensive echo cancellation devices. Also, when the endpoints are physically close—e.g., when you phone your neighbor down the street—the delay is so minimal that anything you transmit will be returned back so quickly that it will be indistinguishable from the sidetone* normally occurring in your telephone.

Why Echo Occurs

Before we discuss measures to deal with echo, let’s first take a look at why echo occurs in the analog world.

If you hear echo, it’s not your phone that’s causing the problem; it’s the far end of the circuit. Conversely, echo heard on the far end is being generated at your end. Echo is caused by the fact that an analog local loop circuit has to transmit and receive on the same pair of wires. If this circuit is not electrically balanced, or if a low-quality telephone is connected to the end of the circuit, signals it receives can be reflected back, becoming part of the return transmission. When this reflected circuit gets back to you, you will hear the words you spoke just moments before. The human ear will perceive an echo after a delay of roughly 40 milliseconds.

In a cheap telephone, it is possible for echo to be generated in the body of the handset. This is why some cheap IP phones can cause echo even when the entire end-to-end connection does not contain an analog circuit.† In the VoIP world, echo is usually introduced either by an analog circuit somewhere in the connection, or by a

* As discussed in Chapter 7, sidetone is a function in your telephone that returns part of what you say back to your own ear, to provide a more natural-sounding conversation.

† Actually, the handset in any phone, be it traditional or VoIP, is an analog connection.

cheap endpoint reflecting back some of the signal (e.g., feedback through a hands-free or poorly designed handset). A good rule of thumb is to keep latency to less than 250 milliseconds.

Managing Echo

In the *zconfig.h* configuration file, you can choose from one of several echo canceller algorithms, with the default being MARK2. Experiment with the various echo cancellers on your network to determine the best one for your environment. Asterisk also has an option in the *zconfig.h* file to make the echo cancellation more aggressive. You can enable it by uncommenting the following line:

```
#define AGGRESSIVE_SUPPRESSOR
```

Note that aggressive echo cancellation can create a walkie-talkie, half-duplex effect. This should be enabled only if all other methods of reducing echo have failed.

Enable echo cancellation for Zaptel interfaces in the *zapata.conf* file. The default configuration enables echo cancellation with `echocancel=yes`. `echocancelwhenbridged=yes` will enable echo cancellation for TDM bridged calls. While bridged calls should not require echo cancellation, this may improve call quality.

When echo cancellation is enabled, the echo canceller learns of echo on the line by listening for it for the duration of the call. Consequently, echo may be heard at the beginning of a call and eventually lessen after period of time. To avoid this situation, you can employ a method called “echo training,” which will mute the line briefly at the beginning of a call, and then send a tone from which the amount of echo on the line can be determined. This allows Asterisk to deal with the echo more quickly. Echo training can be enabled with `echotraining=yes`.

Asterisk and VoIP

It should come as no surprise that Asterisk loves to talk VoIP. But in order to do so, Asterisk needs to know which function it is to perform: that of client, server, or both. One of the most complex and often confusing concepts in Asterisk is the naming scheme of inbound and outbound authentication.

Users and Peers and Friends—Oh My!

Connections that authenticate to us, or that we authenticate, are defined in the *iax.conf* and *sip.conf* files as *users* and *peers*. Connections that do both may be defined as *friends*. When determining which way the authentication is occurring, it is always important to view the direction of the channels from Asterisk’s viewpoint, as connections are being accepted and created by the Asterisk server.

Users

A connection defined as a user is any system/user/endpoint that we allow to connect to us. Keep in mind that a user definition does not provide a method with which to call that user—the user type is used simply to create a channel for incoming calls.* A user definition will require a context name to be defined to indicate where the incoming authenticated call will be placed in the dialplan (in *extensions.conf*).

Peers

A connection defined as a peer type is an outgoing connection. Think of it this way: *users* place calls to us, while we place calls to our *peers*. Since peers do not place calls to us, a peer definition does not typically require the configuration of a context name. However, there is one exception: if calls that originate from your system are returned to your system in a loopback, the incoming calls (which originate from a SIP proxy, not a user agent) will be matched on the peer definition. The default context should handle these incoming calls appropriately, although it's preferable for contexts to be defined for them on a per-peer basis.†

In order to know where to send a call to a host, we must know its location in relation to the Internet (that is, its IP address). The location of a peer may be defined either statically or dynamically. A dynamic peer is configured with `host=dynamic` under the peer definition heading. Because the IP address of a dynamic peer may change constantly, it must register with the Asterisk box to let it know what its IP address is, so calls can successfully be routed to it. If the remote end is another Asterisk box, the use of a `register` statement is required, as discussed below.

Friends

Defining a type as a friend is a shortcut for defining it as both a user and a peer. However, connections that are both a user and a peer aren't always defined this way, because defining each direction of call creation individually (using both a user and a peer definition) allows more granularity and control over the individual connections.

Figure 8-1 shows the flow of authentication control in relation to Asterisk.

* In SIP, this is not *always* the case. If the endpoint is a SIP proxy service (as opposed to a user agent), Asterisk will authenticate based on the peer definition, matching the IP address and port in the Contact field of the SIP header against the hostname (and port, if specified) defined for the peer (if the port is not specified, the one defined in the `[general]` section will be used). See the discussion of the SIP `insecure` option in Appendix A for more on this subject.

† For more information on this topic, see the discussion of the SIP context option in Appendix A.

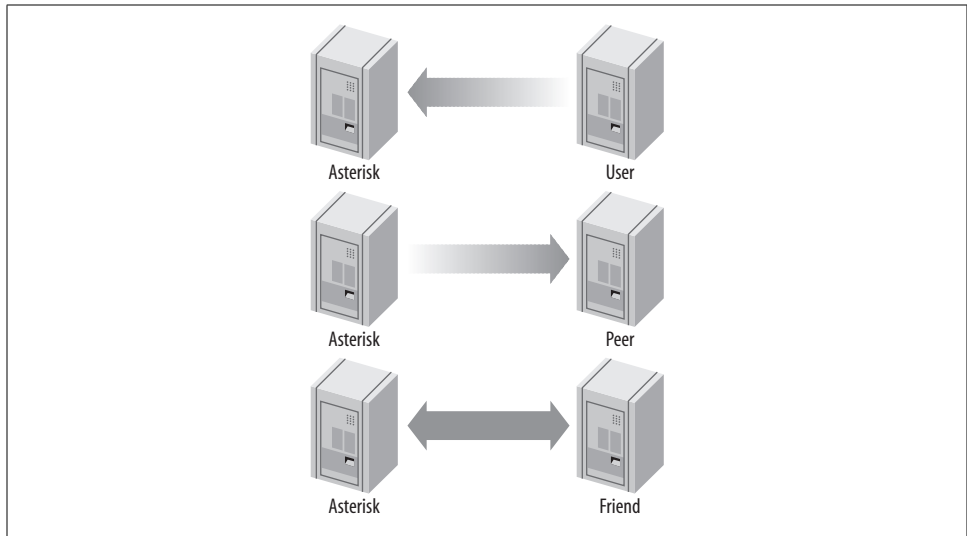


Figure 8-1. Call origination relationships of users, peers, and friends to Asterisk

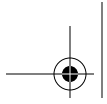
register Statements

A register statement is a way of telling a remote peer where your Asterisk box is in relation to the Internet. Asterisk uses register statements to authenticate to remote providers when you are employing a dynamic IP address, or when the provider does not have your IP address on record. There are situations when a register statement is not required, but to demonstrate when a register statement is required, let's look at an example.

Say you have a remote peer that is providing DID services to you. When someone calls the number +1-800-555-1212, the call goes over the physical PSTN network to your service provider and into their Asterisk server, possibly over their T-1 connection. This call is then routed to your Asterisk server via the Internet.

Your service provider will have a definition in either their *sip.conf* or *iax.conf* configuration file (depending on whether you are connecting with the SIP or IAX protocol, respectively) for your Asterisk server. If you receive calls only from this provider, you would define them as a user (if they were another Asterisk system, you might be defined in their system as a peer).

Now let's say that your box is on your home Internet connection, with a dynamic IP address. Your service provider has a static IP address (or perhaps a fully qualified domain name), which you place in your configuration file. Since you have a dynamic address, your service provider specifies *host=dynamic* in its configuration file. In order to know where to route your +1-800-555-1212 call, your service provider needs to know where you are located in relation to the Internet. This is where the register statement comes into use.



The register statement is a way of authenticating and telling your peer where you are. In the [general] section of your configuration file, you would place a statement similar to this:

```
register => username:secret@my_remote_peer
```

You can verify a successful register with the use of the `iax2 show registry` and `sip show registry` commands at the Asterisk console.

Conclusion

If you listen to the buzz in the telecom industry, you might think that VoIP is the future of telephony. But to Asterisk, VoIP is more a case of “been there, done that.” For Asterisk, the future of telephony is much more exciting. We’ll take a look at that vision a bit later, in Chapter 11. In the next chapter, we are going to delve into one of the more revolutionary and powerful concepts of Asterisk: AGI, the Asterisk Gateway Interface.

