Chapter 9

Asterisk and *app_rpt*

If you've kept up with the major projects in the open-source community, you've probably come across *Asterisk*. The *Asterisk* PBX has earned a reputation as a respectable substitute for the traditional business phone system, which until recently was typically an expensive "black box" without much flexibility. Compared to other phone systems, *Asterisk* has a lot of advantages. It runs on a conventional, inexpensive PC. The *Asterisk* software is free (free as in beer *and* free as in speech), and runs on a free operating system (*Linux*). It is endlessly configurable, offering plenty of built-in flexibility as well as a wide variety of plug-in options. In fact, *Asterisk* is so flexible that it's become a platform for both commercial and Amateur Radio VoIP systems. This chapter describes just such a system.

Unlike some of the other equipment and software described in this book, an *Asterisk*-based VoIP system is more like a toolkit than a ready-to-run machine. If you learn how to use the tools, you can create a system that does exactly what you need. But learning the ins and outs of the toolbox might take some time, especially if you haven't fiddled with open-source software very much, so expect to devote some extra effort to an *Asterisk*-based VoIP system. You might find that it's time well spent, since you're likely to learn plenty about *Linux* along the way, and end up with a custom-designed solution that works exactly the way you want it to.

Flexibility is the biggest advantage of the "toolkit" approach. With a system built around *Asterisk*, you can build a node that's part of a larger, public VoIP network, or you can set up a private VoIP link for a special purpose. One example is a set of point-to-point links that join two or three remote receivers in a wide-area repeater system. Another might be a remote-base-style system that lets you jump on a distant repeater using a low-power handheld transceiver around the house. **Figure 9-1** shows some of the possibilities for Internet linking using *Asterisk*.

A VOICE SWITCH

A PC running *Asterisk* is a modern version of the clicking, whirring mountain of relays and circuit boards known as a *private branch exchange*, or PBX. The PBX is essentially a switch that can route telephone calls between an office full of telephones and a set of lines provided by the phone company. The PBX lets workers in the office make calls to each other, as well as incoming and outgoing calls to and from the outside world.

The job of a PBX is a little more complicated these days, since telephone calls are

carried over a combination of voice and digital circuits — the conventional twisted pair (sometimes called POTS, an acronym for Plain Old Telephone Service) and VoIP. A computerized switch is a natural choice for VoIP, since all of the routing and switching can be handled in software. It can also work with POTS lines, as long as it's equipped with adapters that can translate between analog and digital signals.

In *Asterisk*-speak, optional software modules are called *applications*, with short, unpronounceable names beginning with *app_*. The key application for Amateur Radio linking on *Asterisk* is *app_rpt*. The *app_rpt* application sits between the *Asterisk* core and the hardware interface to the radio equipment, handling radio-related functions such as carrier detect, frequency control, and transmitter control (PTT).

The *app_rpt* application is the centerpiece of a system called TIARA, which

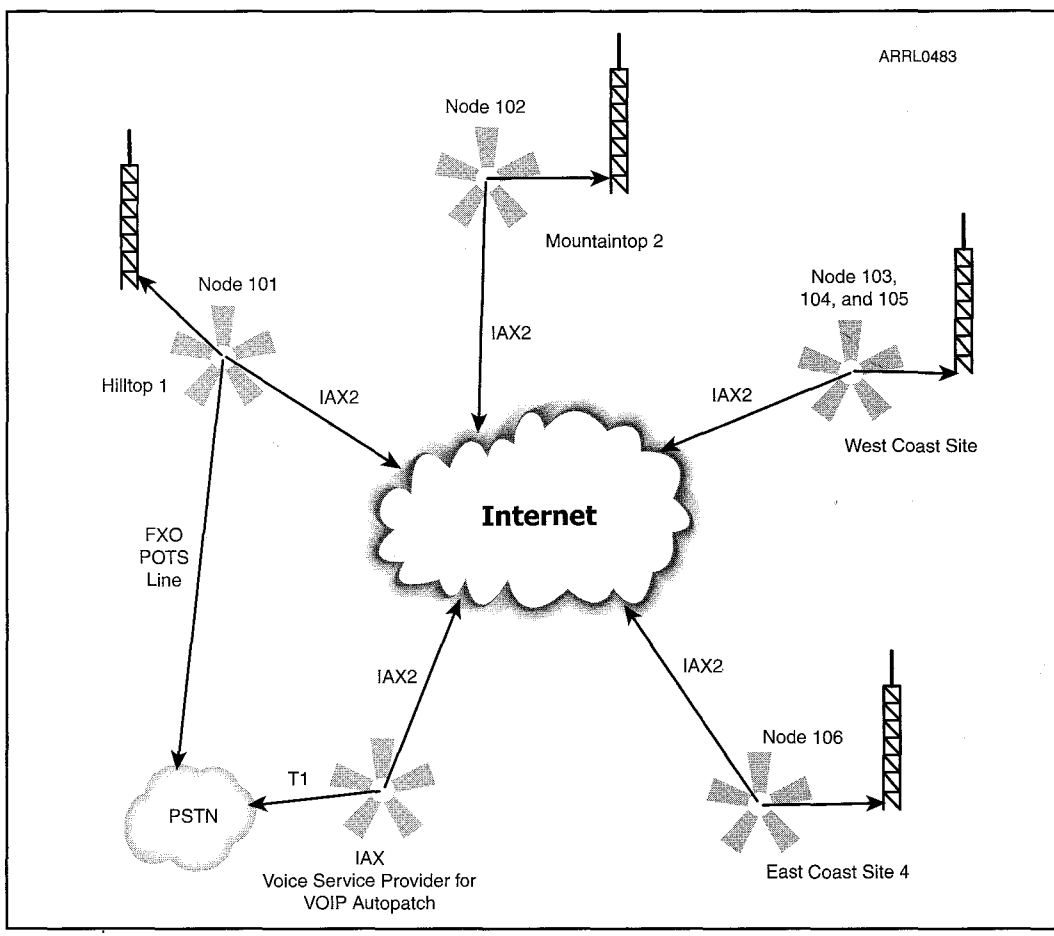

Figure 9.1 — The *Asterisk* system with *app_rpt* provides plenty of possibilities for Internet linking, as well as full-featured repeater control.

stands for "Technology Implemented by *App_Rpt / Asterisk*." The TIARA project aims to provide a whole host of radio-related functions on the *Asterisk* platform. With the right configuration, the system can be used as a full-blown repeater controller, a remote-base controller, or a VoIP Internet link. TIARA can take advantage of *Asterisk's* software-based DTMF decoder to handle commands sent over-the-air or from an incoming phone call, and remote control of the system over the Internet. Since all of the software is open source — from the *Linux* operating system all the way up — anyone familiar with programming can jump in and make custom changes, improvements, and enhancements.

NODES, LINKS, AND CHANNELS

It's important to remember that when you're building a repeater linking system around *Asterisk* and *app_rpt*, you're creating a complete repeater controller, not just an Internet link. The system can either be operated by itself as a stand-alone repeater, or connected to over the Internet from other nodes that are similarly equipped.

Unlike a typical repeater controller, an *Asterisk*-based system has all of the repeater control functions running in software, rather than hardware. Just as when *Asterisk* is functioning as a telephone switch, virtually all of the audio that travels through the system is handled digitally, rather than as analog signals. It's the job of the interface device to convert analog signals to and from digital, and this happens at the earliest and latest possible points along the path. The big advantage of this technique is that changing the node's "personality" is just a matter of updating its software, rather than having to make difficult and potentially costly hardware changes. It also means that very complicated switching and bridging arrangements can be set up quickly and easily, again without touching the hardware.

Another important term in the *Asterisk* vocabulary is *channel.* Channels are the endpoints of communication: a telephone set, a telephone line, or a radio transceiver, as examples. Each different kind of channel has a separate software module (called a *channel driver*) that "knows" how to handle a certain technology. Inside the *Asterisk* software, two or more channels can be bridged to one another, establishing a connection that allows two-way communication. Each type of channel has a configuration file that sets up the options for the channel, and specifies how certain kinds of events on the channel should be handled.

The next piece of the puzzle is the *dialplan*, which is a kind of recipe or roadmap for how the various channels should interact. Although the term is a carryover from the world of telephones, not every item in a dialplan is something that actually needs to be "dialed," nor do any of the channels have to be telephone-related. Instead, a dialplan can describe (for example) what should happen when a VoIP connection request is received from another node over the Internet. *Asterisk* has a rich set of commands that can be invoked in the dialplan, somewhat like a computer programming language.

INTERFACING

For connecting radio equipment to *Asterisk*, there are a couple of options. The first-class approach is the Quad PCI Radio Interface, available from QRV Communications (**qrvc.com**). As the name implies, this is an adapter that fits inside the computer in a peripheral component interconnect (PCI) slot, and provides interface ports for up to four transmit/receive devices (**Figure 9-2**). Each port has an audio input, an audio output, a COS line, and a PTT line. The audio input and COS lines connect to a receiver, and the audio output and PTT lines connect to a transmitter. Of course, if a transceiver is being used, all four lines connect to the transceiver, since it is providing both functions at once.

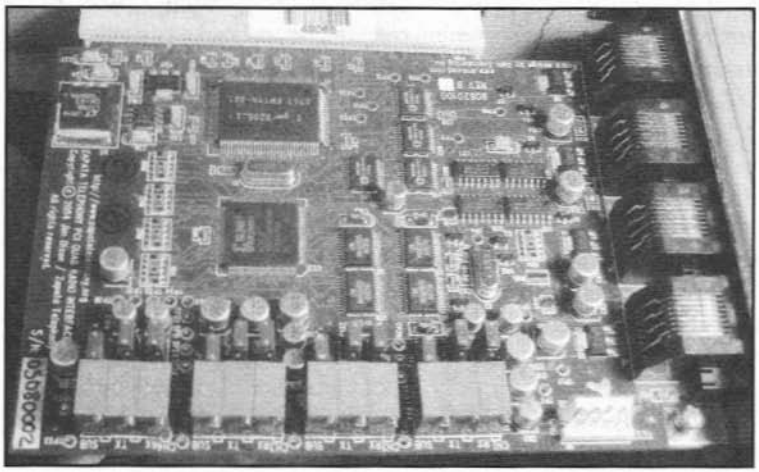

Figure 9.2 — The Quad PCI Radio Interface.

Another option is a device called the USB Radio Interface (URI), affectionately known as the USB Fob (**Figure 9-3**), or the Dongle. The URI (from DMK Engineering, **dmkeng.com**) connects to the PC's USB port and provides connections to a single transmitter/receiver. It is built around a single chip, the C-Media CM108, that acts as a USB sound "card" with extra input/output lines that are used for the PTT and COS connections. A companion channel driver lets the URI play nicely with *Asterisk*. With this arrangement, DTMF digits received over the air are decoded in software using *Asterisk's* own DSP-based decoder.

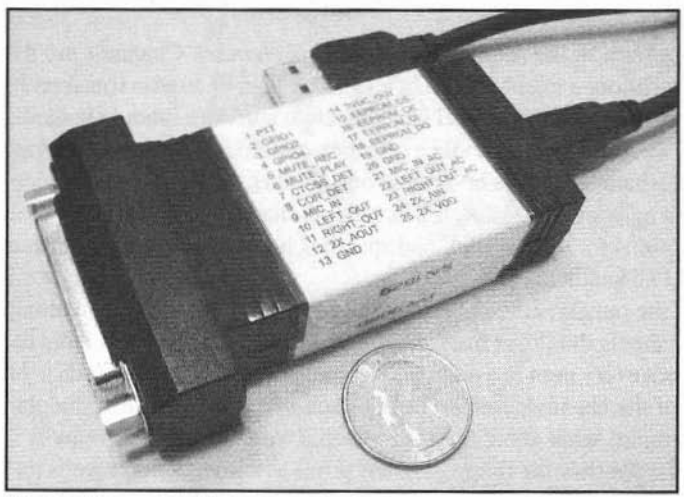

Figure 9.3 — His Eminence, the Fob.

THE ALLSTAR LINK SYSTEM

If you've set up an Amateur *Asterisk* system with *app_rpt*, you're invited to add your node to the list of nodes on the AllStar Link network. Just as with other linking systems such as *EchoLink* and *IRLP*, each node on the network is assigned a node number, and nodes can be connected to one another by using DTMF commands on a portable or mobile radio within range of the node's repeater or remote base.

An AllStar Link node requires some extra software to join the network. The easiest way to set up a complete *Asterisk/app_rpt* system for the AllStar system is to download and burn a single CD-ROM that includes a copy of the CentOS distribution of *Linux* and a complete *Asterisk* installation. This downloadable image is available from the AllStar Link Web site (**allstarlink.org**). The Web site also has information about how to request an AllStar Link node number.

One key difference between a repeater node on the AllStar Link system and a repeater node on another system such as *EchoLink* or *IRLP* is that the *Asterisk* box functions as both the repeater controller and the Internet link — the two functions are completely integrated into a single controller. In contrast, *EchoLink* and *IRLP* repeater nodes are typically either hard-wired to a dedicated repeater controller, or placed at a location separate from the repeater and joined to the repeater itself over the air. You can think of a system like the AllStar network as a worldwide collection of repeater controllers that live on the Internet.

Choosing a PC

Just as with other kinds of linking systems, it's worth giving some advance thought to what kind of PC you will use to host the system. For an *Asterisk/app_rpt* installation, this consideration is even more critical, since the PC will be hosting the repeater itself, not just the Internet link. Reliability is the number-one consideration; as long as the machine has sufficient memory and minimally adequate CPU speed, it will hum along just fine; the latest top-of-the-line dual-core screamer is probably not the best choice.

If the site is not easily accessible, consider using an embedded PC that is rack-mountable and can run from a single-sided 12-volt power supply. Just one PCI slot is required (if using the PCI Quad Radio Interface card), so you can easily use a small machine that draws modest power and generates minimal heat. Some embedded PCs have no moving parts at all (no fan or hard drive), leaving them with many fewer things that can go wrong in the middle of the night at the top of a mountain in the dead of winter.

INSTALLATION

There are several ways to set up an *app_rpt* machine for use on the AllStar Link network, but the easiest is to download and run the Allstar CentOS Install Disk (ACID). ACID is a complete image of the CentOS *Linux* distribution, along with the latest *Asterisk* package, *app_rpt*, and the appropriate configuration files. You can download

the ACID CD-ROM ISO image from the **allstarlink.org** Web site, and burn it on to a blank CD. Be sure to also download and print a copy of the installation instructions and the System Administrators guide; these will walk you through the installation and configuration steps to get you up and running quickly.

But before running the installer, if you're expecting to add your node to the AllStar network, you'll need an AllStar link node number. Node numbers can be requested from the **allstarlink.org** Web site. When you run the ACID installer, you'll be prompted to enter your new node number and password near the end of the installation process. The install script automatically stores this information in the appropriate *Asterisk* configuration files before it starts up *Asterisk* for the first time. There is also an option to run a small program that reports the status of your node to the **allstarlink.org** Web site periodically.

Once the installation is finished and the machine is running with its newly-configured software, the next step is to connect and adjust the equipment. For either the Quad PCI interface card or the URI, you'll need to put together a custom-made cable to connect the radio gear to the *Asterisk* interface. The Quad PCI interface has RJ-45 (modular) connectors, and the URI has a 25-pin computer-type connector. For some radio-interface combinations, you may need a few additional components (such as switching transistors and resistors) mounted on a small circuit board between the interface and the rig. For example, the URI has an open-collector output for the PTT line that goes to ground when the PTT is activated; if your equipment expects the reverse of this sense, you'll need to add a simple inverter stage between the URI and the transmitter. The documentation for either interface has detailed pin-outs and schematics, which should be followed carefully.

The main adjustment step is to set the audio levels to and from the radio. Although you can do a rough adjustment by ear, the most accurate deviation adjustments require a service monitor. The *Asterisk/app_rpt* software packages have functions that will generate test tones at specific reference levels to make this easier. Audio-level adjustment might involve a combination of hardware and software adjustments; in particular, the Quad PCI interface has multi-turn trimpots for each of the four radio ports directly on the PCI card. All of the audio adjustments for the URI device are handled in software.

LINKING FROM A PC

The *Asterisk* package supports several different standard VoIP protocols, such as Session Initiation Protocol (SIP). Thanks to standards such as SIP, it's possible to use various non-*Asterisk* programs and devices to connect over the Internet to an *Asterisk*-based system. For example, you can use an IP phone, which resembles a conventional telephone, but is actually a complete digital voice terminal with an Ethernet connector instead of a conventional twisted-pair port. Or, you can use any number of desktop PC applications that emulate the functions of an IP phone.

But there's a catch: These IP devices and programs are designed to work like

conventional, full-duplex telephones. They're not a good fit for the Amateur Radio world, which is predominantly half-duplex. What's missing is a push-to-talk switch — a convenient way to take control of a half-duplex link to make a brief transmission.

Fortunately, there's a desktop PC application called *iaxRPT* that solves this problem. IAX stands for Inter-Asterisk eXchange (pronounced "eeks"); it's an Internet protocol specifically designed to connect *Asterisk*-based systems to one another. The *iaxRPT* program, which is offered as a free download from **www.xelatec.com/xipar/iarpt** can connect to a remote system running *Asterisk* and *app_rpt* using IAX2 to carry on a remote QSO. At first glance it resembles a typical "soft phone" application, but it also handles the all-important PTT function, and it's built expressly for working with *app_rpt*.

When setting up your *Asterisk/app_rpt* node, the installer creates a couple of configuration files on the machine (iax_rpt_custom.conf and extensions_rpt_custom.conf) that set up the server side of the link. When you install *iaxRPT* on another computer, you'll be prompted to enter the matching information (node number and password, for example) to let you connect remotely to your own node. Once everything is up and running, you can hook up to your repeater or remote base from anywhere in the world, and carry on a QSO using the desktop computer's microphone and speakers. The **CTRL** key on the keyboard works as the push-to-talk switch; press it to begin speaking, release it to stop.