Installing and Configuring Asterisk

Exchanging Phonecalls

Asterisk offers a low-budget approach to IP telephony, putting this technology within the price range of small to mid-sized businesses. The Open Source software based switchboard links up PCs, internal hardware-based IP phones, and voice mailboxes with PST and mobile networks. BY THORSTEN SPÄTH

www.photocase.de

Low prices for broadband connections make IP telephony systems an attractive proposal. If you can do without a normal telephone system, using Open Source software on a Linux server instead, you can save a lot of cash into the bargain. One of the best examples of this is a software phone system called Asterisk [1]. Asterisk is a mature product and scales well from mini-networks to large-scale installations. You will need DSL, or something faster for your connection, to ensure that you can talk to normal PST phone systems without any drawbacks.

This workshop will be focusing on configuring Asterisk as a VoIP exchange for a small network, and setting up an answering machine and a holding loop. Asterisk is not a IP-only solution. The PBX software also allows you to talk to analog or ISDN phones via SIP.

THE AUTHOR

Thorsten Späth has been working with Linux-based business solutions since 1996. His job involves implementing Linux for small to medium-sized businesses all over the country.

The Asterisk developers are making rapid progress. Modules are modified and bugfixes added on a daily basis [2]. On the downside, some CVS versions are unreliable. Once you have a stable version installed and running, the recommended approach is to leave well alone, unless you are forced to update.

Small Business Scenario

A small business is looking to send an ISDN telephone system to the happy hunting grounds, allowing its staff to use IP telephones. They have chosen Asterisk to route internal calls via the LAN, or use an Internet connection in case of external calls to staff. An ISDN card in the Asterisk server provides a failsafe should the Internet link fail.

There are three groups within the business: the boss is out of the office quite often, and the best way to get in touch with him is by cellphone. Two members of staff work at the company offices, and there are two more that work from home, dialing up the company via ISDN or DSL, which makes them attainable through VoIP also.

The five-person company has a local network based on 10/100 Mbit Ethernet, and a 2 MBit sDSL leased line (see Figure 1). The offices are also equipped with

Listing 1: *sip.conf* – phone definition

```
01 [sales]
02 type=friend      ; Default settings: User is both user and peer
03 secret=password   ; password for phone
04 fromuser=sales   ; username for phone
05 username=sales   ; username for phone
06 dtmfmode=rfc2833         ; keytones after dialing, as per RFC
07 host=dynamic     ; user sales can be mobile with dynamic IP
08 context=default ; The phone will reside in the default extension
   context
09 canreinvite=no   ; for hardware compatibility reasons
```

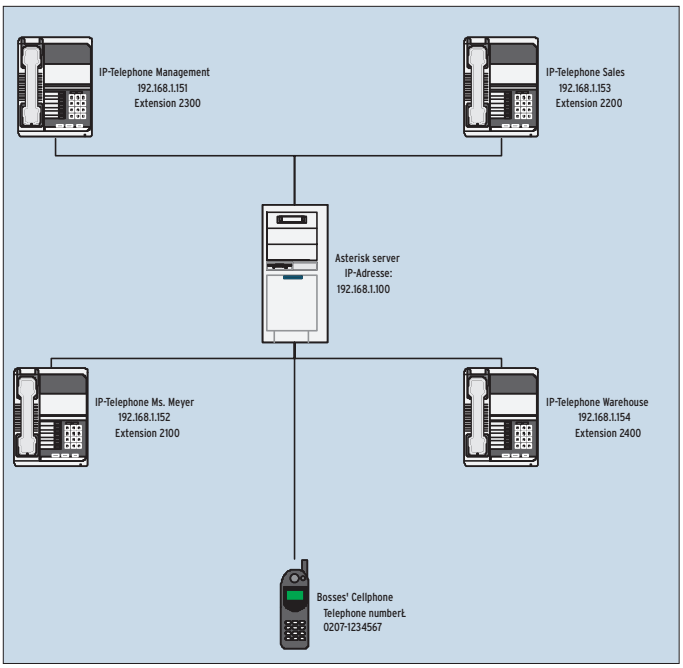

Figure 1: Our chosen scenario features a network in a small business without a firewall or NAT.

two parallel-wired ISDN basic rate (S0) ports, thus providing a total of four ISDN data channels (B channels). The connection to the outside telephony world is routed via the VoIP provider Nikotel [3], which has a worldwide voice network based on SIP (Session Initiation Protocol). The provider also routes calls into the PST or cellphone networks.

Configuration

Setting Asterisk up is non-trivial as it involves numerous configuration files. It makes sense to take notes as you proceed, allowing you to retrace your configuration steps. At this time of writing, Asterisk does not have a working Web or GUI-based front-end, although a front-end is under development.

The PBX software stores the data channels for ISDN and VoIP in separate files below */etc/asterisk/*. ISDN telephones are defined in *modem.conf*, IP telephones in *sip.conf*, which also contains the setup for connecting to the IP telephony provider.

sip.conf is the best place to start the configuration. You need to assign a unique ID to each IP phone and user. The sections with the users in our sample file all start with a suitable mnemonic. The configuration for the user *sales* is shown in Listing 1. You need to add all of your users and phones.

The settings for the keytones after connect (tone dial system) and the NAT-specific parameters are important. NAT is normally a good thing, but it does make configuring SIP and H.323 telephony quite difficult (see the box "Problems: NAT and VoIP"). Check out the search feature provided by the Voip-Info wiki at [10] for tips on NAT and firewall configurations.

After creating an entry for the first IP telephone in *sip.conf*, you need to turn your attention to the VoIP provider data. Our example uses Nikotel [3]. Minor changes are required for other VoIP providers. The IP telephone answers to the internal network number 991234512345. The SIP-Provider uses this number for the internal assignment of the user data, which is represented by the real user name along with the other access data in *sip.conf*.

You need to modify the values for *localnet* and the netmask in Listing 2 to reflect the values your IP interface uses. This takes care of the preparatory steps; we have registered an account, and configured our first IP telephone.

Call Routing

Configuring Asterisk as a switchboard is more complex. The Asterisk control file is called */etc/asterisk/extensions.conf*. It provides a central point for routing incoming and outgoing calls. The idea is to tell the PBX software to route external callers directly to internal extensions, and to allow internal IP extensions to place outside calls via the VoIP provider.

extensions.conf is subdivided into individual sections. The Asterisk daemon interprets the file sequentially from the top downward. Sections can use the

Problems: NAT and VoIP

Most users with DSL at home configure their routers to use NAT. This can cause major issues with the UDP-based SIP standard. NAT uses a single public address to support multiple internal machines. The SIP standard does not support NAT address translation, and this means that the phones will use incorrect IPs in packet headers.

Most SIP providers use a workaround involving a STUN server to avoid this problem. Once a STUN directory entry exists, packets will use the IP provided by STUN to synchronize. Things start to get tricky if your hardware has a firmware problem that you are not aware of. Although the system seems to get a line, the dial tones are not forthcoming. This may be a router issue if your router fails to allow RTP (Real Time Protocol) traffic to pass. In this case, you need to map the router ports explicitly.

Convincing NAT and Asterisk to talk to each other in a sensible and stable manner can involve a lot of tinkering. Hiding your Asterisk server behind a firewall makes things even more complicated. At least for a start, you may want to keep everything on a single local subnet, to provide an elegant, although temporary, workaround.

Required Packages

It is a good idea to run Asterisk on a system with kernel 2.4, as a few issues occurred with the PBX software when we tried to run it on a kernel 2.6 machine. Besides Asterisk, you require the following packages: *libpri* [4], and *zaptel* [5]. Also make sure that *OpenSSL* [6], and *readline* [7] (including the source packages) are installed.

To avoid tripping up over the daily Asterisk snapshot releases, it is a good idea to install the current stable release. You can either download it from the CVS (see [8] for a howto), or use FTP. The top level directory for our installation will be */usr/src/*.

```
cd zaptel
make clean; make install
cd ../libpri
make clean; make install
cd ../asterisk
make clean; make install
```

If something goes wrong at this stage, there is plenty of support available via IRC, the mailing list at [9], or in the Voip-Info [10] wiki for any stage of the installation or configuration.

included keyword to include other sections. The *[default]* section is used to assign a section to a user.

By default, any calls from the SIP network will be handled by the procedure in the *[incoming]* section. This is where you tell Asterisk how to deal with incoming calls. The demo data supplied with the Asterisk package show you how to do this. External callers hear a welcome message first, and are then told that the call will be passed to a member of staff. Listing 3 shows the configuration for the *[incoming]* section.

When a call is routed to the system via *nikoteldial*, we want Asterisk to pick up the virtual receiver first (*1,Answer*). While the software is routing the call to an internal extension on the distribution list, the caller is treated to some canned music in the form of a GSM-formatted sound file, *demo-congrats.gsm*. Line 4 uses *Goto* to put the call through to *[DifferentSection]*, for example.

In this *[DifferentSection]* the call is computed otherwise for dealing with special issues like putting all incoming calls through to a queue (see Listing 4). If an error appears, the local daemon hangs up. This approach is fine for initial tests, as we do not want callers to talk to the answering machine at this point. We can take care of the answering machine, as soon as we get the virtual switchboard working.

Asterisk defines these steps as so-called extensions within the framework of a dial plan (plan and actions for incoming and outgoing call routing. The syntax abbreviates the word extension to *exten*.

Each extension has three parameters. The first parameter describes the extension number or the alias (*nikoteldial*).

The second specifies the priority within the current context. Finally, Asterisk expects you to define an action, or point it to an application. In configurations with large numbers of extensions, you could use multiple Asterisk systems to route calls across network boundaries.

Listing 2: *sip.conf* – Account data

```
01 [general]
02 port=5060
   ; SIP server port
03 bindaddr=0.0.0.0
   ; required for multihost
   (multiple IPs)
04 externip=123.123.123.123
   ; External IP address (for
   NAT)
05 localnet=192.168.1.0
   ; local subnet (important for
   NAT)
06 localmask=255.255.255.0
07 context=incoming
   ; the context for incoming
   calls
08 disallow=all
   ; first disable all codecs,
   then
09 allow=ulaw
   ; allow codecs in the
   following order
10 allow=alaw
   ; ulaw (G.711 for USA), alaw
   (G.711 for Europe)
11 register =>
   username:password:@calamar0.ni
   kotel.com:5060/nikoteldial
```

Switchboard Practicalities

Of course, it is no big help to dump callers after leaving them on hold for a specified period. To avoid this, calls are first sent to a distribution list where an another member of staff can hopefully handle them.

Callers are first treated to a spot of canned music. While this is playing, Asterisk rings all the phones in the distribution list.

We still need to handle cases where the call is not picked up by a member of staff. There are several possible approaches. If the caller is trying to contact Sales, it would be useful to say something like "All of our lines are busy, please call back later!", or to ask the caller to leave a message on the answering machine.

To route the call through the distribution list, we need to replace the *Goto* statement in line 3 of Listing 3 with the following :

```
exten => s,3,Queue(holdloop)
```

The distribution list is defined in *queues.conf*. This is where you can specify the maximum hold period, the extensions to be called, and the maximum backlog in the queues. Listing 4 shows how to do this for four IP phones.

Asterisk retrieves music on hold from a default pool in */var/lib/asterisk*. It uses a subdirectory for music on hold, voice mail messages, call management, and welcome messages. Check out the "Using your own sounds" box for details on changing the default sounds.

The */var/lib/asterisk* directory has a number of subdirectories and files containing scripting elements for CGI and PHP programming with Asterisk (AGI). You can also use Asterisk as a server to distribute your telephone firmware. Sadly, all of this is beyond the scope of our minimal installation.

Routing Calls to the Answering Machine

Now that all incoming calls are first sent to the hold loop, you need to deal with situations where a caller can not be put through to a member of staff within the three minute hold period. An answering machine is your last line of defense in this case. You can automatically notify a member of staff by email and voice mail when a caller leaves a message on the answering machine. Of course, Asterisk allows you to call the answering machine and listen to the message.

To enable the answering machine feature, the admin user needs to add an extension, and assign a mailbox number to Sales. Asterisk stores voice messages

Listing 3: *extensions.conf* – Incoming

```
01 [incoming]
02 exten => nikoteldial,1,Answer
   ; Pick up phone
03 exten =>
   nikoteldial,2,Background(demo-
   congrats)     ; Welcome
   message
04 exten =>
   nikoteldial,3,Goto(otherSectio
   n,s,1)        ; Jump to
   "otherSection"
05 exten => nikoteldial,4,Hangup
   ; Hang up
06 exten => i,1,Playback(invalid)
   ; Invalid procedure, retry
07 exten => t,1,Hangup
   ; Hang up if nothing else
   happens
```

in */var/spool/asterisk/voicemail/default* by default, sorting the messages by phone number and any caller information such as the date or a call back number.

The answering machine details, and corresponding actions, are stored in *sip.conf*,

```
[sales]
....
mailbox=2200
```

extensions.conf:

```
...
exten => nikoteldial,1,➥
Voicemail(u2200)
```

and *voicemail.conf*:

```
...
[default]
2200 => 1234,sales,➥
sales@mycompany.com
```

The first step is to assign a voice mailbox to Sales. The line with *mailbox = 2200* takes care of this. This extra line must follow the statement that redirects calls to the hold loop. Make sure you keep the lines of the configuration file in the right order.

The *[default]* section in *voicemail.conf* defines the voice mailbox feature, assigning the mailbox *2200* to the user *sales*, and protecting the box with a password, *1234*. The system forwards incoming messages to an email address *sales@mycompany.com*.

Setting Up Asterisk for Outgoing Calls

To allow Ms. Meyer, your Sales department, and any other members of staff, to place outgoing calls, you need a few more definitions in the *[default]* context

Using your own sounds

The default sounds supplied with the Asterisk distribution are fine for an initial configuration. After you get the initial configuration running, you might like to replace them with your own sounds. If you prefer more individual music for the hold loop, you can use MP3 files (or other formats). Mpg123 [11] is fine as the player. Be careful that you do not infringe against any copyright laws if you use commercial tracks!

You need to store your MP3 collection in */var/lib/asterisk/mohmp3*. By default, Asterisk will search through this directory and allow *mpg123* to play the audio files in random order. After recording your own messages, store them in GSM format below */var/lib/asterisk/sounds*. Asterisk expects these files without the gsm extension, for example *transfer* for *transfer.gsm* in a message such as "Your call is being transferred".

If you do want to use your own voice, or prefer to use a female voice as a male reader, try the high quality synthetic voice from AT&T Labs [12]. Failing this, you can always contact a service agency to have an actor or actress put your message across in a friendly way. In contrast to many other low-budget systems, Asterisk is quite happy with a loss free digital recording – and that is a good thing!

of your *extensions.conf* file (see Listing 5). Enter a channel for ISDN calls, and one for calls via the SIP provider here. In our example, we will be defining a speed dialing shortcut for the boss's cellphone.

Lines 2 and 3 assign Ms. Meyer's phone and the phone in the Sales department the numbers 2100 and 2200. Whenever an internal caller picks up an IP phone and dials *2100*, the Asterisk server steps in and rings Ms. Meyer. Some phones, such as the Grandstream Budgetone, allow you to enter a # sign directly after dialing the number, to avoid the timeout.

The system can use either an IP telephony service provider, or the ISDN card to place outgoing calls to the PST network. Callers need to use the 99 prefix to select ISDN dialing. A prefix of 98 is used to route calls via the SIP provider. This tells the PBX server what channel to use for the connection.

If the call can not be placed within 20 seconds, the system plays a file called *Invalid Extension* and then hangs up. Let's assume that Ms. Meyer wants to call an extension on the SIP network. For Nikotel, she would need to dial 98991234567890. 98 identifies the SIP network; this is followed by the extension number assigned by Nikotel, 991234567890. You can simplify this long dialling number by defining shortcut dialing rules.

Listing 4: *queues.conf*

```
01 [holdloop]
02 music=default           ; Use
   default hold text or canned
   music
03 strategy=ringall        ; Call
   strategy: ring all the
   extensions on the list
04 timeout=15              ;
   Assume unavailability after 15
   second wait
05 retry=3                 ; Try
   three times
06 maxlen=5                ; Hold
   a maximum of five callers in
   the incoming queue
07
08 ; Define four IP phones with
   extensions where they can be
   reached internally
09 member => SIP/2100
10 member => SIP/2200
11 member => SIP/2300
12 member => SIP/2400
```

Using an ISDN Adapter

The ISDN adapter links up the state-of-the-art IP system to the digital past. You need a working ISDN4Linux. Although an alternative driver is available, installing the driver caused a few issues in our testing. To add an ISDN system to your Asterisk setup, you need to enable the appropriate channel in your *modem.conf* file. Listing 6 shows a default setup.

Asterisk has a whole bunch of additional configuration files that allow you to customize the system to reflect various network environments and needs. The *voip-info.org* wiki is a useful reference for admins. It provides numerous sample configurations, explanations of the individual parameters, and tips on specific hardware.

The Asterisk sponsor Digium [13] also has special analog telephone adapters in its portfolio that allow you to use various analog telephones with Asterisk. You need to modify the *zapata.conf* configuration file to add these PCI cards to your system.

Listing 5: Permitting outgoing calls

```
01 [default]
02 exten =>
   2100,1,Dial(SIP/meyer@meyer,60
   ,Ttr)
03 exten =>
   2200,1,Dial(SIP/sales@sales,60
   ,Ttr)
04 ; Bosses' Cellphone via 6666
05 exten =>
   6666,1,Dial(Modem/ttyI0:020712
   34567,20,r)
06 ....
07 ; Use 99 to select ISDN
   channel
08 exten =>
   _99.,1,Dial(Modem/ttyI0:${EXTE
   N:1},20,r)
09 exten =>
   _99.,2,Playback(invalid)
10 exten => _99.,3,Hangup
11 ; Use 98 for SIP-based calls
12 exten =>
   _98.,1,Dial(SIP/${EXTEN:0}@nik
   oteldial,20,r)
13 exten =>
   _98.,2,Playback(invalid)
14 exten => _98.,3,Hangup
15 ....
16 exten => h,1,Hangup
```

Launching Asterisk

Now that you have modified the required configuration files, and integrated a few phones, there is nothing to prevent you from launching Asterisk. It makes sense to launch your virtual switchboard in a very verbose mode, to have as much information as possible on the initialization procedures:

```
asterisk -vvvvc
```

This sends a whole bunch of messages across the screen. The following prompt then appears:

```
CLI>
```

Congratulations, Asterisk is running! You can enable various debug modes to check if the software is working correctly. Type *help* in the command line for more details.

The *sip debug* is typically required, at first. It enables debug output for communication via the SIP channel. This is particularly useful if the IP phones are

Listing 6: *modem.conf* – Enabling ISDN

```
01 [interfaces]
02 context=remote
03 ; Use Isdn4Linux driver
04 driver=i4l
05 type=autodetect
06 ; Enable tone dial
07 dialtype=tone
08 ; View line as "stable" after
   first dialing tone
09 mode=ring
10 group=1
11 ; Accept calls on MSN 123456
   only
12 incomingmsn=123456
13 ; outgoing MSN 123400
14 msn=123400
15 ; Asterisk assigns the ISDN
   channels
16 device => /dev/ttyI0
17 device => /dev/ttyI1
```

not working, or the SIP provider you configured does not react. The debug messages include a complete set of headers, and this can be vital if you need to troubleshoot a NAT network. As soon as you have everything running to your satisfaction, you can quit debug mode by entering *sip no debug*.

Another *sip* command provides admins with user information. You can type *sip show peers* to display active users, and *sip show users* to view the user configuration.

A Long Way to Go

Asterisk is one of the most powerful freeware telephone systems, period. The phone system function is just one of Asterisks features. You can use Asterisk as a VoIP gateway, an exchange, or even for providing telephony services. It connects the world of traditional phone systems with the more modern world of TCP/IP. Besides a steadily increasing user base and improved hardware support, Asterisk is slowly but surely becoming a one-stop communication hub.

At this time of writing, administrators are still required to follow the traditional Linux approach to installation, and this involves modifying numerous configuration files. Web gateways for PHP with MySQL and AGI interfaces are under development. This should allow non-expert users, after completing the basic installation steps, to individually configure extensions, recording complex messages for answering machines and adding music on hold. At its current stage of development, Asterisk does need a lot of tinkering and pioneering work. On the upside, the fact that the CVS build is updated daily shows that there are a lot of developments to look out for in future.

Asterisk is quite difficult to configure at present. However, once you become used to the nomenclature, you will grow to appreciate the Asterisk system, which can hold its sway with commercial and proprietary solutions. If you are thinking of using Asterisk in a production environment, don't forget that Asterisk can only be as stable as your hardware. Silence may be golden, but not in your office, if your VoIP machine crashes. ■

INFO

[1] Asterisk: *http://www.asterisk.org*

[2] Bugfixes: *http://bugs.digium.com*

[3] Nikotel: *http://www.nikotel.de*

[4] Libpri: *ftp://ftp.asterisk.org/pub/telephony/libpri/libpri-0.6.0.tar.gz*

[5] Zaptel: *ftp://ftp.asterisk.org/pub/telephony/zaptel/zaptel-0.9.1.tar.gz*

[6] OpenSSL: *http://www.openssl.org*

[7] Readline: *http://cnswww.cns.cwru.edu/~chet/readline/rltop.html*

[8] Asterisk CVS: *http://www.asterisk.org/index.php?menu=download*

[9] Mailing lists: *http://www.asterisk.org/index.php?menu=support*

[10] VoIP-Info: *http://www.voip-info.org*

[11] Mpg123: *http://www.mpg123.de*

[12] AT&T Sounds: *http://www.research.att.com/projects/tts/demo.html*

[13] Digium: *http://www.digium.com*