ROBERT SOLOMON

a 36-user Asterisk installation

Phone System Administrator is one of several hats Bob Solomon wears at a medium-sized non-profit in New York City. He would like to participate in more Asterisk projects.

bobsol@gmail.com

MY NON-PROFIT EMPLOYER NEEDED to replace an aging Toshiba KSU system because the phones, some of which date back to the '80s, were falling apart and the voicemail's hard drive was grinding loudly. I was able to convince management to replace the system with an Asterisk-based PBX by promising a lower end cost and the elimination of vendor lock-in. In this article, I walk you through the process I followed: selecting phones, modifying the Asterisk configuration, and training users.

Asterisk installations of the sort I had done previously, where Asterisk functioned as an answering machine or a small office phone system, have been documented in many places. This installation offered an opportunity to take my Asterisk skills to the next level. The system supports eight lines, 24 extensions, and over 36 voicemail users.

Selecting Phones

Originally, I proposed a system utilizing a channel bank and Aastra 9116 phones. The final design used Polycom IP430 and IP601 SIP phones. The keypad and voice quality of a sample 9116 was disappointing in comparison to our existing Toshiba EKT phones.

Business-quality analog phones are available at about the same price point as the Polycom SIP phones that we ultimately selected. To support the SIP phone the only hardware required is an Ethernet or, ideally, Power Over Ethernet port. The per port cost of POE is much less than that of an FXS (analog phone station) port.

The existing phone system provided voice-first intercom; that is, intercom calls were answered by the recipient's phone on speaker phone, without any action on the part of the recipient. A warning tone preceded the call so that the recipient knew when he or she no longer had privacy. An all-page feature similar to voice-first intercom, but connecting to most phones on the system at once, one way, was also provided. A conversation with management confirmed the importance of retaining these features. I tested a sample Polycom IP430. This phone could handle voice-first calls with better sound quality than the existing system. The keypad feel and sound quality compared favorably with our existing phones.

Polycom phones can be configured from the keypad, through an HTML interface or by XML files, or automatically downloaded by the phone at bootup. I strongly recommend setting up the Polycom phones to download their configuration from the Asterisk server rather than configuring the phones individually. Getting the phones talking with the server's dhcpd, (VS)ftpd, and ntpd was a day's work [1].

Infrastructure and Hardware

The Toshiba system used twisted pair, but SIP phones require CAT5 or better. I considered and rejected the use of our data network for telecommunications.

The separate voice network I constructed incorporated a POE switch, eliminating the need for an AC adapter at each phone. This is convenient for the users and makes the installation of wall phones away from an outlet cleaner and easier. The voice-only network avoided bottlenecks in the existing data network and segregated SIP phone registration and ftp configuration traffic, enhancing security.

I selected a Netgear 24-port POE switch, model FS728TP, based on POE watts available and price. The fan in this switch was noisy enough that someone in the adjoining room complained, so I removed the switch from the rack-mount and placed it on sound insulating material on a rack-mount shelf.

Asterisk is hosted here on a server built on an Asus TS300-E4/PA4 with a Xenon dual-core 3070 2.66-GHz processor with 1 GB RAM. Call volume is about 6,800 calls a month. I can rip a CD on the system for music on hold during times of heavy call traffic without degradation in voice quality.

Connection to eight POTS lines (FXO) and four FXS ports for analog extensions is provided by a set of three Sangoma A200 cards with optional hardware echo cancellation. Echo cancellation is always needed with modern analog (really digital) voice cards. Hardware echo cancellation is of better quality than freely available software offerings, reduces the load on the CPU, and saves several hours of time tuning settings to eliminate echo. Only one echo cancellation module is needed per set of Sangoma cards. There have been no complaints about sound quality on the calls handled using these cards except for rare reports of echo when someone is using a headset. This is resolved by turning down the volume on the extension. Another advantage of the Sangoma cards is that they do not require a PCI slot with a unique IRQ per four lines as did the Digium TDM400P cards used on a previous system. Setting up three TDM400Ps with unique IRQs can take hours on the wrong motherboard.

The Digium cards have been redesigned and I have no experience with the new version, TDM410P, which addresses these issues. The Sangoma cards have noisy FXS or station ports. Most of our extensions are SIP phones, so the noise has not been a problem here.

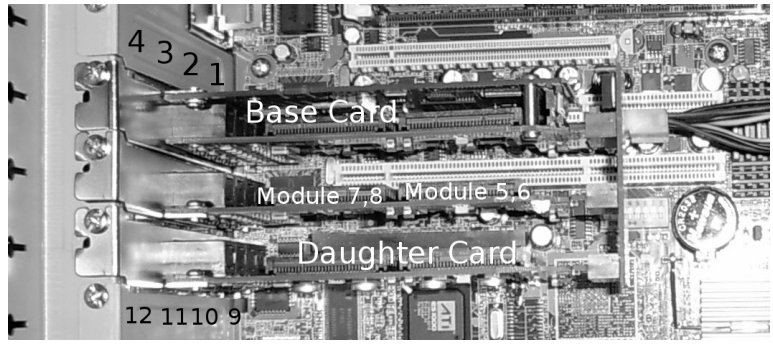
Hardware Issues

About three weeks after the system went live, while I was away for the weekend, the system crashed. When I checked the log files, I found that the system had crashed at a time when no calls were active. An updatedb cron job had been running. Within a week, the system crashed again early in the morning while running a backup. Initially I suspected a failing hard drive. When the system was taken offline, late at night, memtest86 was run on a hunch and memory failure was detected [2]. There have been no crashes

since the memory was replaced. Please note that the symptoms of this memory failure were freezes during disk I/O and md5sum sometimes reporting bad check sums on good files.

The server was temporarily moved to a physically smaller SATA-only system that had no Molex power connectors. I spent the night trying to get more than one Sangoma card to work in the replacement server and learned some details about the Sangoma A200 card system that are not obvious from the sales literature, although they are more or less documented on the Sangoma wiki hardware page [3].

Sangoma A200 cards are assembled into a system consisting of a base card, daughter cards, and modules using a small backplane. As many as 24 ports can be installed using one PCI slot. In the case of this system, 12 ports are provided by a base card, two daughter cards, and six modules (see Figure 1). This assembly of cards uses one PCI slot, but it fills three openings in the back of the box. An advantage to Sangoma's system compared to others that use separate PCI slots for each group of modules is that all channels have common synchronous clocking [4].



That night, I tried installing the cards in many different configurations and combinations, but each time I rebooted the server, wancfg_zaptel would detect the cards, but loading the modules with "wanrouter start" would fail when more than one card was installed. This was because the Sangoma backplane requires a 12-V connector and the power it supplies even if no FXS (station) cards are installed.

Ports are numbered according to the backplane slot used, with the lowest ports in the leftmost slot. If a slot is skipped, there will be a gap in the port numbers.

The base card can be plugged into any socket in the Sangoma backplane. Clearance between the cards is acceptable with the base card in any position but is best when the base card is in the leftmost socket on the backplane.

The next day a co-worker suggested that I locate and remove the defective memory stick and switch back to the proper server. Service was restored to all ports while I waited for replacement memory to arrive.

Extensions Using Dialplan Pattern Matching

I used pattern matching in the dialplan for calls to the extensions, rather than a macro. In the global section of the dialplan a variable like the ones shown for extensions 12 and 13 is set for each real extension. To add an extension, all that must be done to the dialplan is to add another variable like those here:

```
x12=Sip/12
x13=Zap/11 ;door phone
```

The following dialplan excerpt handles intercom calls placed from an inside context:

```
; line below is for voicemail for calling self
exten => _[1-8]X,1,GotoIf($[${CALLERID(num)} =
  ${EXTEN}]?CheckVoiceMail)
; check if there is an extension if not go to voicemail
exten => _[1-8]X,n,GotoIf($[!${EXISTS(${x${EXTEN}})}]?Voicemail)
exten => _[1-8]X,n,SIPAddHeader(Alert-Info: Ring Answer) ; voice first
exten => _[1-8]X,n,Dial(${x${EXTEN}},20,tk)
exten => _[1-8]X,n,Voicemail(${EXTEN},u)
exten => _[1-8]X,n,GotoIf($[${VMSTATUS} = FAILED]?NoVoicemail)
exten => _[1-8]X,n,Hangup()
; "this extension has no voicemail or your message was
; too short"
exten => _[1-8]X,n(NoVoicemail),Playback(cust89)
exten => _[1-8]X,n,Hangup()
exten => _[1-8]X,n(CheckVoiceMail),VoiceMailMain(${EXTEN})
exten => _[1-8]X,n,Hangup()
```

If the extension matches the caller ID, we check voicemail. The Polycom phones place a call like this when voicemail is called from the voicemail button on the phone. If an extension variable does not exist, we go straight to voicemail, caller side. If voicemail fails, an error message is played to the caller.

Tuning the System

The Polycom phones were dumbed down and customized. Call forwarding was disabled in the site Polycom configuration file to prevent abuse by our users and guests.

```
<divert>
  <fwd divert.fwd.1.enabled="0" divert.fwd.2.enabled="0"
    divert.fwd.3.enabled="0" divert.fwd.4.enabled="0"
    divert.fwd.5.enabled="0" divert.fwd.6.enabled="0"/>
</divert>
```

Call waiting was turned off for each extension in sip.conf:

```
call-limit=1
```

Calls to the 601s, used by our receptionists, still came through on call waiting. I had to add the following line to the Polycom site configuration file to get these phones to return a busy signal:

```
<call call.callsPerLineKey="1">
```

This was important because I wanted calls that come through while the receptionist is talking to come through on the additional line buttons rather than as call waiting calls on the first button.

A DND (Do Not Disturb) hard key was added to every Polycom 430. I edited the site phone configuration file and physically changed a key cap on every phone:

```
<keys key.IP_430.32.function.prim="DoNotDisturb" />
```

To set up voice first I edited the Polycoms' configuration and the Asterisk dialplan. The site-wide Polycom configuration file was modified, providing a ring class with a ring type of ring-answer [5, 6]:

```
<alertInfo voIpProt.SIP.alertInfo.1.value="Ring Answer"
  voIpProt.SIP.alertInfo.1.class="4"/>
<RING_ANSWER se.rt.4.name="Ring Answer" se.rt.4.type="ring-answer"
      se.rt.4.timeout="1000" se.rt.4.ringer="11" />
```

In the dialplan a SIP header must be sent to the phone on a per-call basis to make that call voice first:

```
exten => #30,1,SIPAddHeader(Alert-Info: Ring Answer)
```

Users initiate an all-page by pressing #30, the code used on the previous system. In the dialplan, this is implemented with the Page command:

```
exten => #30,n,Page(${ALL_PAGE})
```

A context for blind transfers is provided and the __TRANSFER_CONTEXT variable set to avoid the blind transfers going through ring-answer: When testing the system, before this change was made, a transferred call would be automatically answered and on speaker phone whether the intended recipient was at her desk or not.

```
exten => s,n,Set(__TRANSFER_CONTEXT=t_c) ; context for blind transfer
```

I have not yet found an equivalent solution for supervised transfers. These transfers are currently set up to go through voice first. Ideally, the first contact, introducing the transfer, would go through voice first. When the receiving party accepts the call and the transferring party hangs up, the transferee's phone should ring.

The Polycoms and Asterisk do not play well where call parking is concerned. The Polycoms expect the user to provide the parking space number; Asterisk expects to provide the space number. To use the park feature key on the Polycoms, the user has to press "more" then "park," press a bogus extension number that Asterisk will ignore, and then press park again. If the phone has more than one line presence, Asterisk calls the user back with the parking space number, rather than just announcing the number. Users universally elect to park calls PBX style, dialing a two-digit feature code on the keypad rather the using the feature on the phone. In this case, all the user has to do is dial *2 and the system immediately parks the call and announces the parking space number to the user.

The two-digit park sequence is set in features.conf. I needed to adjust the interdigit timeout to 1,000 ms, a value that works for our users:

```
[general]
...
;featuredigittimeout = 1000
[featuremap]
...
parkcall => *2  ; Park call (one step parking)
```

The park key on the Polycom phones requires a callpark extension in the dialplan.

```
exten => callpark,1,ParkAndAnnounce(silence/1:pbx-transfer:
      PARKED|120|SIP/
${DIALEDPEERNUMBER}|internal,${DIALEDPEERNUMBER},1)
```

In either case, parkedcalls must be included in the context:

```
include => parkedcalls
```

Extension Aliasing

When an employee leaves the organization, I am often asked to forward that employee's calls to the extension or mailbox of the person taking over the departed employee's workload.

Initially, a documented solution could not be found [7]. Experimentation revealed that this can be done with a Goto:

```
exten => 69,1,Goto(74,1); 69 is an alias for 74
```

Door Intercom and Door Lock Control

The old system provided an intercom with a door lock control at the side door. To implement this feature with Asterisk, I provided a Viking analog speaker phone, model E-20B, with no dial and an auto answer feature. This phone looks like an intercom. Calls from this phone start in an immediate context in the Asterisk dialplan and ring selected extensions automatically.

```
[door]
; door intercom
exten => s,1,Answer()
exten => s,n,Dial(${DOOR_CALLS},30,tk)
exten => s,n,Playback(silence/1)
exten => s,n,Hangup()
```

The immediate keyword is set in zapata.conf so that the s extension executes without any action other than going off hook when the intercom user presses "call":

```
context=door
immediate=yes
callerid="Door Intercom"
group=2
signalling = fxo_ks
channel => 11
```

A Viking door control box, C-2000A, providing a relay for lock control, and a MIS1C DTMF relay board available from Mike Sandman were considered for door lock control. I selected the Sandman board because connection is in parallel with the phone on an available FXS, station, port. The C-2000A requires an FXO (central office) port and costs four times as much as the MIS1C. The C-2000A includes a metal case and offers many features that I did not need here.

To open the door, users press a programmable code while on the door intercom call [8]. I added the following line to the incoming context:

```
exten => 13,1,Goto(13,NoVoicemail)
```

When callers press the door intercom number, 13, from an outside call they hear "This extension has no voicemail." Callers are prevented from calling the door intercom from outside the building.

Managing Day, Night, and Holiday Mode Changes

Asterisk configuration is finer-grained then the proprietary systems I have worked with in the past. The area of day and night call handling provides a good example of this.

The system plays a different main message depending on whether we are open, closed for the night, closed for a holiday, or closed on an August Sun-

day. Also, calls to the operator go directly to voicemail when we are closed for any reason. Management stipulated that these changes in day and night call handling be automated.

Once per incoming call a global variable MODE is read from the persistent Asterisk database:

```
exten => s,n,Set(GLOBAL(MODE)=${DB(vars/MODE)})
```

The value of this variable has been set to 0 through 3, with 0 representing open and the positive integers representing various closed states. A main message is chosen based on the value of the variable:

```
; set and play the main message day|night|holiday|sunday-in-August
exten => s,n,Set(MSG2PLAY=cust02)
exten => s,n,ExecIf($[${MODE} = 1]|Set|MSG2PLAY=cust03)
exten => s,n,ExecIf($[${MODE} = 2]|Set|MSG2PLAY=cust07)
exten => s,n,ExecIf($[${MODE} = 3]|Set|MSG2PLAY=cust08)
exten => s,n,Background(${MSG2PLAY})
```

If MODE is true when a caller in the incoming context presses 0 a night message is played and the Operator extension does not ring:

```
exten => s,n,GotoIf(${MODE}?oper-night,s,1) ;play night if mode > 0
```

A cron job, changing the value of MODE in the Asterisk DB, is run at closing time every day.

```
# night mode, weekday, saturday
30 20 * * mon-fri,sat /usr/sbin/asterisk -rx \
  'database put vars MODE 1' &>/dev/null
```

At opening time in the morning, cron runs a Perl script that checks for a holiday and changes the value of MODE accordingly [9]. Luckily, we are open for business on Easter.

Advantages to this approach are persistence of the MODE variable over a restart of Asterisk or even a reboot and control of our schedule with cron and an easily maintainable Perl script. Holiday determination is made once per day rather than once per call.

In the event of a mishap, the mode can be changed from any phone, as follows:

```
[set-mode]
; "system is in day|night|holiday mode"
exten => s,1,Background(cust9${DB(vars/MODE)})
; "press 0 for day, 1 for night, 2 for h..."
exten => s,n,Background(cust88)
exten => s,n,Waitexten(5)
exten => s,n,Hangup()

exten => _[0-3],1,Set(DB(vars/MODE)=${EXTEN})
; "system is in day|night|holiday mode"
exten => _[0-3],n,Background(cust9${EXTEN})
exten => _[0-3],n,Hangup()

exten => i,1,Background(pbx-invalid)
exten => i,n,Goto(s,1)
```

Training Users

The previous KSU phone system provided presence for six of our outside lines at each phone. When the organization was smaller, with only three in-

coming lines, a KSU setup was advantageous because call handling was intuitive. We have had eight lines for many years now and Asterisk provides for better call handling than could be provided with line buttons on each phone. Because PBX call handling requires skills that will be new to most users, training is required for receptionists and others in the front line of call handling.

I reached an understanding with management about the need for user training prior to implementation. Management's cooperation in this area was key to the success of the project.

Training was provided for about half of our employees one on one and in small groups. I trained some staff myself and also trained others to train. A dialplan extension was provided which moves a call from the inside to the outside context for the purposes of training and testing. Trainings took place at a desk with two extensions and a cell phone.

Some useful training material was available on the Web, particularly at the site of the Woods Hole Oceanographic Institution (WHOI). Unfortunately, these pages have been moved or removed. I customized the material on the WHOI site and wrote additional material [10].

We use four techniques for handling calls here: supervised and blind transfers, call parking, and exclusive hold.

Blind and supervised transfers were new to our users. In a supervised transfer, the transferee speaks to the recipient before the call is connected. For nonemergency, nonexecutive calls, a blind transfer to the recipient's extension is the best way to handle a call.

On the old system, a call could be placed on hold and a wanderer paged: "So and So, please pick up a call on line three." Users are intimidated by call parking, even though the process is very similar. When a call comes in for a wanderer, the call is parked and the recipient paged: "So and So, please pick up a call on 701."

I got repeated questions about how to forward a voicemail. When I tried this myself, I found that upon completion of a successful voicemail forward the system says, "Your message has been saved." The user is left to wonder if the transfer has gone through. This prompt can't be changed because the system uses it for other purposes.

Is an Asterisk System Right for Your Site?

A bare-metal Asterisk installation offers savings in hardware costs over a proprietary system but adds costs for the time spent in configuration. Packaged four-line, eight-extension turnkey small business phone systems (without phones) are available for under $2,000 [11, 12]. Some of these systems are Asterisk-based. The server, switch, voice cards, and 24 phones used here cost $7,200. Components related to cabling and infrastructure cost $2,100. Twelve weeks of labor went into this project, including planning and research, requisition, premises wiring, system installation, configuration, and initial training of staff.

Asterisk must be able to provide the features expected by management and users with a reasonable amount of work. Voice-first intercom was a must-have feature here. Two-digit extension numbers and some feature codes that have been in use for over two decades were preserved. A door intercom was provided. Phones needed to be simple and intuitive to use.

You should have a good idea of what a properly working phone system sounds and feels like. Experience administering the existing or another similar system is a big plus. Compassion for the user experience is important. In a small to medium-sized business, cabling skills may be handy.

The sustainability of a phone system that is managed by a system administrator editing files in an organization this size concerns me. I have not investigated available Asterisk GUI front ends, under the assumption that these front ends would not be capable of the degree of customization needed here. Secondary personnel should be trained in the administration of the system so that changes and failures can be addressed when the primary administrator is not available.

Conclusion

Despite concerns about sustainability and a few too many rough edges, this Asterisk installation has been successful. Users and management are happy with the system. Sitting at the console and watching three dozen people interact daily with a system running on Patrick Volkerding's Slackware is satisfying.

Resources

The Asterisk beginner would do well to start with the book *Asterisk, the Future of Telephony* [13]. At the Asterisk console, man style information on any dialplan application or function can be accessed by typing:

```
fone*CLI> core show application ApPlIcAtIoN ;not case sensitive
```

or:

```
fone*CLI> core show function FUNCTION ;must be uppercase
```

When I need more detail than either of these resources provides, the voip-info.org Asterisk pages [14] are often helpful. Start with the voice board vendor's Web site for installation instructions for both the kernel modules supporting their boards and Asterisk itself. In the case of the Sangoma boards used in this project, the Sangoma wiki [15] was very helpful. The Asterisk Web site [16] provides a good overview of the Asterisk project. If Polycom SIP phones are used don't fail to download the *Administrator's Guide for the SoundPoint IP/SoundStation IP Family* [17].

REFERENCES

[1] http://www.voip-info.org/wiki/view/Asterisk%40Home+Handbook+Wiki+Chapter+7#7221WhychoosePolycomVOIPPhonesbspan.

[2] http://weaversrevenge.com/ast36/burned-out.jpg.

[3] http://wiki.sangoma.com/sangoma-hardware#A200.

[4] http://www.sangoma.com/products_and_solutions/hardware/analog_telephony/.

[5] http://www.voip-info.org/wiki/index.php?page=Asterisk+cmd+page.

[6] See p. 151 of the *Administrator's Guide for the SoundPoint IP/SoundStation IP Family:* http://www.polycom.com/common/documents/support/setup_maintenance/products/voice/SoundPointIP_SoundStationIP_AdminGuide_SIP3_0_Eng_Rev_A.pdf.

[7] http://asterisk-jtapi.sourceforge.net/setup.html.

[8] http://weaversrevenge.com/ast36/door-buzzer.

[9] http://weaversrevenge.com/ast36/holidays.

[10] http://weaversrevenge.com/ast36/voicemail-crib.odt.

[11] http://shop.talkswitch.com/details/CTTS001184001.asp?.

[12] http://store.digium.com/products.php?category_id=41.

[13] J. Van Meggelen, L. Madsen, and J. Smith, in *Asterisk: The Future of Telephony,* Second Edition (Sebastopol, CA: O'Reilly Media, 2007): http://www.asteriskdocs.org/.

[14] http://www.voip-info.org/wiki/index.php?page=asterisk.

[15] http://wiki.sangoma.com/.

[16] http://www.asterisk.org/.

[17] http://www.polycom.com/common/documents/support/setup_maintenance/products/voice/SoundPointIP_SoundStationIP_AdminGuide_SIP3_0_Eng_Rev_A.pdf.