DiMi TELECOM

Asterisk Open Source PBX/iPBX Advanced Usage

Presented by:
Nir Simionovich
DimiTelecom Ltd

About Dimi Telecom

- Established 2002, Dimi Telecom has been operating in the Telecom Market as a VoIP retail & wholesale provider.
- Since 2003, Dimi Telecom has been heavily involved in the Asterisk and GnuGK Open Source projects.
- Dimi Telecom is currently Digium's Gold Reseller in Israel.
- Dimi Telecom currently operates Tier-1 installations of Asterisk based appliances.

What are we going to talk about?

- Voice Over IP Primer
  - Signaling Protocols
  - Codecs
- Voice Over IP and Asterisk
  - H323 (chan_h323)
  - SIP (chan_sip)
  - IAX (chan_iax2)
- The NAT Traversal problem
- The Asterisk Application Gateway Interface (AGI)
  - AGI Programming Basics
  - AGI Simple Examples
  - AGI Execution Flow and the DeadAGI concept
- Questions and Answers

Voice Over IP – VoIP for short

A Primer to Voice Over IP

- Voice Over IP (VoIP) can be described as the ability to sample voice transmissions, packet them into well known chunks, and transmitting them over an IP network.
- VoIP traffic is split into 2 network transmission types: Signaling and RTP. Signaling is usually TCP based, while RTP is UDP based.
- Signaling is usually performed between well known TCP ports (h323:1720, SIP:5060), while RTP uses randomly assigned UDP ports.

VoIP Signaling: H323

- H.323 is the international standard for multimedia communication over packet-switched networks, including LANs, WANs, and the Internet. It was first defined by the ITU in 1996 and has been updated regularly. The most recent version is H.323 version 5 (2003).
- H.323 was the world market leader for transporting voice and video around the world, with literally billions of minutes of voice/video traffic every month alone.
- H.323 is a binary stream based protocol, which means that interoperability between vendors is hard, at times even impossible. This is caused by the various H323 version implementations of vendors.

VoIP Signaling: Session Initiation Protocol (SIP)

- The Session Initiation Protocol (SIP) is the IETF standard for the establishment of multimedia sessions. These sessions might be used for audio, video, IM, or other real-time data communication sessions.
- The scope of SIP is relatively broad, including the establishment of virtually any kind of "session" between two parties. SIP is also entirely independent of the underlying transport, though TCP and UDP are used almost exclusively.
- Unlike its counter part, H323, SIP isn’t a binary stream signaling protocol. SIP signaling is very similar to the HTTP protocol standard, which makes it highly inter-operable between vendors.
- Although H323 is the more common one, SIP is slowly becoming a more dominant option.

VoIP Signaling: Inter Asterisk Exchange (IAX)

- IAX is the de-facto standard VoIP protocol for Asterisk networking.
- Perhaps its most impressive feature is its transparent interoperation with NAT and PAT (IP masquerade) firewalls, including placing, receiving, and transferring calls and registration.
- IAX is extremely low-overhead (four bytes of header, as compared to at least 12 bytes of header for RTP based protocols like SIP and H.323). IAX control messages are also substantially smaller.
- IAX supports internationalization, permitting the requesting PBX or phone to receive content from the providing PBX in its preferred language if available.
- Supports authentication on incoming and outgoing calls (Border Control).
- Using IAX dialplan polling, the dialplan for a collection or cluster of PBX's can be centralized, with each PBX only needing to know its local extensions.
- Unlike its counterparts, IAX can support trunking, giving a better performance per Kbps ratio.

Codecs: Voice Coders, what's available and for what price?

Asterisk provides internal support for the following codecs:

Codec	Kbps	Quality
G711alaw	64 Kbps	Phone
G711ulaw	64 Kbps	Phone
Linear 16bit	128 Kbps	Phone
ADPCM	32 Kbps	Poor quality
GSM 6.10	13 Kbps	Cellular Phone
LPC-10	2 Kbps	Mr. Roboto
iLBC	13 Kbps	Cellular Phone

Asterisk also supports g729 via purchase of a licensed codec module, either from Digium or directly from VoiceAge. G723.1 is supported as pass-through only.

H323: chan_h323 and chan_oh323

- H323 support is not native to Asterisk, and is provided via external channel modules.
- The 2 available modules are: chan_h323 provided by Jeremey McNamera and chan_oh323 provided by InAccess Networks.
- Both channels are based upon the OpenH323 stack.
- Both channels use the same RTP stack, provided by Asterisk.
- chan_h323 is slightly easier to install than chan_oh323, which requires various patches to the OpenH323 stack.
- chan_h323 provides better control for incoming and outgoing call control. chan_oh323 reminds very much the Cisco manner of managing VoIP dial-peers.
- Both channels are in constant development and testing, and are known to work in production environments around the world.
- Use what ever you feel more comfortable with!

H323: Known issues

- As H323 is a byte stream based signaling protocol, and it's currently at version 4, inter-operability issues may occur.
- chan_h323 had been known to cause problems with some versions of Cisco IOS, while chan_oh323 is known to cause problems with other versions.
- As support for H323 is rendered via externally shared libraries, both channels suffer from an instability factor.
- The instability would usually end up in one of the following scenarios: Asterisk crashes with a nice core-dump to send to the programmers, Asterisk becomes non-responsive due to channel dead-locks.
- In the latter case, Asterisk will remain responsive to the CLI, but will not handle any calls.
- Personal Recommendation: use only if you must, avoid if not needed.

SIP: chan_sip

- SIP support is developed as a standard channel that is provided with Asterisk.
- In comparison to H323, SIP is much more stable and inter-operable.
- SIP inter-operability had been achieved with Cisco, SNOM, GrandStream, Veraz, Sonus and more vendors.
- Asterisk supports SIP as a SIP registrar or a SIP agent.
- With the availability of SIP phones everywhere, SIP is becoming the protocol of choice for iPBX installations.
- Asterisk supports SIP clients that are located behind a NAT or a PAT network.
- The Asterisk SIP stack can operate behind a NAT firewall, seamlessly.
- Real-Time configuration of peers and clients is available via an internal resource.

SIP: chan_iax2

- IAX is the de-facto standard for Inter Asterisk Exchange.
- IAX is a proprietary signaling protocol, and does not inter-operate with other vendor devices.
- IAX soft phones and IAX VoIP phone are available.
- IAX support is developed as a standard channel that is provided with Asterisk.
- Asterisk can operate as an IAX registration server or as a SIP UA.
- IAX is operates seamlessly behind NAT and PAT.

The issue with NAT traversal

- Both SIP and H323 suffer from a lack of proper support for NAT traversed networks.
- The problem is sourced at the fact that both protocols rely on dynamic port allocation for RTP transmission (Voice).
- H323 handles the issue of NAT traversal with H323 Proxy servers. SIP handles NAT traversal via utilizing STUN servers, or via firewall piercing techniques.
- The above mentioned solution cause problems when debugging configuration in case of problems.
- NAT traversed networks make up for about 95% of the office and home networks around the world – for that NAT/PAT is a serious problem when deploying VoIP networks.

Asterisk H323: NAT traversal handling

- Both chan_h323 and chan_oh323 DO NOT handle any form of NAT traversal handling.
- For H323, NAT traversal can be achieved by utilizing an externally installed GateKeeper in Proxy mode.
- The GnuGK project (www.gnugk.org) can operate as a NAT Proxy for H323.
- The GateKeeper is required to be installed on the network itself.
- The solution is hard to manage, and would usually prove un-reliable.

Asterisk H323: To GK or not to GK

Using an H323 gatekeeper is not a must, but should we use one?

- An H323 gatekeeper as a NAT proxy device for an H323 enabled VoIP network.
- When your H323 gateway doesn't support complex routing rules well.
- When the number of routing rules is too great for a gateway to handle.
- When several gateways need to share the same termination and origination ports.

In any of the above situations, using an H323 gatekeeper would prove as a worth while practice, and would lower your overall system administration efforts.

Asterisk H323: Gatekeeper Registration (LRQ, LCF and LRJ)

- As a rule of thumb, when utilizing a gatekeeper to maintain several endpoint connections, the practice of having the endpoints register to the gatekeeper is good. By utilizing this method, all endpoints will share the common gatekeeper routing policy, while the gatekeeper has full knowledge of the location of each H323 endpoint.
- In order for a registered endpoint to send a call via the gatekeeper, it must first issue an LRQ (Location Request) to the gatekeeper. If the destination appears in the routing policy, it will respond back with an LCF (Location Confirm) or an LRJ (Location Reject) in a case where the destination is not in the policy.

Asterisk H323: Gatekeeper as an IP2IP Gateway

- Another popular method of operation is to use a gatekeeper as a static IP2IP gateway.
- In an IP2IP gateway mode, each termination endpoint is statically mapped in the gatekeeper routing table.
- The incoming endpoints should be authenticated via an external AAA mechanism.
- In this case, the usual manner of LRQ/LCF method is not utilized.
- Call setup is performed utilizing ISDN Q.931 messages.

Asterisk SIP: NAT traversal handling (Asterisk Server behind NAT)

- Asterisk supports a situation when it operates behind a static NAT firewall.
- The proper signaling port should be opened on the firewall (TCP 5060), and also access to the high-udp-ports should be opened (UDP 1025-65534).
- As part of the global SIP configuration of Asterisk, the real routable IP address of the Asterisk server should be defined.

```
[general]
port = 5060                ; Port to bind to
bindaddr = 0.0.0.0         ; Address to bind to
externip = 200.201.202.203    ; Address that we're going to put in SIP messages if we're behind a NAT
context = default          ; Default for incoming calls
srvlookup = no             ; Enable SRV lookups on outbound calls
;pedantic = yes            ; Enable slow, pedantic checking for Pingtel
tos=lowdelay
realm=dimitel.com
```

- Once that is performed, Asterisk will replace all indications of the binded IP address, with the external IP address defined.

Asterisk SIP: NAT traversal handling (SIP UA Firewall Piercing)

- Firewall piercing is performed when a SIP UA connects to a remote SIP server, and keeps the connection open at all times, in order to facilitate a pre-known RTP port.
- Firewall piercing is not available on all SIP UAs, however, most of them do have this function.
- Is this a good solution? As a general practice, no. This type of configuration will increase the load on the state tables of your firewall.
- When you have multiple SIP UAs in your internal network, the proper way would be to establish a local SIP registrar/proxy, and have a single point of contact to the outside SIP gateway.

Generating outgoing calls via VoIP channels

- The Asterisk dial application can support any of the installed channel modules.
- In order to activate the dial application on a specified channel, the channel type must be defined on the extension line. Eg:

```
exten => _054.,1,Dial,H323/${EXTEN}@192.168.0.1
exten => 101,1,Dial,SIP/${EXTEN}
```

- Once Asterisk identifies the channel type on the dial extension, it will invoke the required channel.
- If the specified channel includes an IP location, Asterisk will attempt to perform a direct dial to that IP number, with the information provided.
- If an IP location is not provided, Asterisk will check it's internally registered VoIP UAs and peers, and will route the call to the proper UA/peer accordingly.

Asterisk VoIP: H323 vs. SIP vs. IAX

It is a clear fact that Asterisk supports VoIP protocols for UA connections is various methodologies. Selecting the proper methodology or a combination of ones is entirely dependent on the actual case.

While H323 is field proven for interconnecting with VoIP carriers, SIP and IAX out perform it in the LAN and UA field.

Our experience with interconnecting UA's and VoIP carriers, had resulted in the following 'rules of thumb':

- UA's should interconnect with an Asterisk server utilizing SIP or IAX.
- Interconnecting with VoIP carriers should be done utilizing SIP or H323.
- If the VoIP carrier supports IAX interconnect, that is the best method of interconnecting.

Asterisk VoIP: User Agents Clients PC Soft Phones Software

- Windows Based:
  - X-Lite and X-Pro from X-Ten networks (in my opinion, the best SIP UA around).
  - Firefly from FreshTel networks.
  - SJPhone from SJ labs.
- Linux Based:
  - KPhone, a SIP phone for the KDE environment.

More SIP User Agents for the PC (Windows/Linux/Mac) can be found at http://www.voip-info.org

Asterisk VoIP: User Agents Clients Hardware Phones

- Cisco: Cisco 7960,7940 and ATA186 SIP phones (Very expensive).
- SNOM: SNOM manufactures SIP phones which are fully featured, and are competitive in pricing.
- GrandStream: GrandStream manufacture budget SIP phones and adaptors.
- More are available at http://www.voip-info.org

AGI – Application Gateway Interface

What is AGI?

- AGI stands for “Application Gateway Interface”
- AGI is the means how application programmers can implement external logics utilizing Asterisk.
- AGI is an application level API, which means that application developed using AGI are not bound by the Asterisk License and are not required to be re-distributed with the source.
- AGI is very much similar to a CGI interface, where communications between Asterisk and your AGI program is performed via an STDIN/STDOUT interface.
- AGI programs can be written in any programming language.

How to execute an AGI program from the Asterisk dial plan?

- Each item in an extension is of the form:

  exten => extension-number,priority,application,arguments

  To launch an AGI script the application is 'agi' and the argument is the filename of your script.

- The script:
  - must be executable
  - must be located in /var/lib/asterisk/agi-bin
  - must be specified complete with an extension

- For example to run a Python script named 'test.py' then a suitable extension item would be:

  exten => 1,2,agi,test.py

Passing arguments to your AGI program

- In order to pass arguments to your AGI program, you need to define them in the extension.
- In example, to launch an AGI program with a parameter would look like so:

```
exten => 1,2,agi,test.php|${CALLERID}
```

- In the above example, Asterisk will pass the environment variable ${CALLERID} to the AGI program test.php

Things to remember about using arguments and AGI programs

- AGI scripts *always* receive two arguments. The first argument is the full path to the script itself. The second argument is the stuff passed in from the "exten" line
- If no argument is given on the "exten" line or if the argument given is empty, then the argument received is an empty string
- The argument received consists of everything on the line following the vertical bar up until a vertical bar, a semi-colon or a comma. That means the argument may contain spaces
- Quotes, single or double, are simply taken as part of the argument; they have no special effect
- By the time you get the argument any trailing spaces have been deleted but leading spaces are not deleted
- **SMALL TIP**: If you want to pass more than a single argument in the argument, simply create a tokenized string, and parse the string inside your AGI program

Communicating with the Asterisk AGI interface

- The communication method between your AGI program and Asterisk is very simple
- Your AGI program should write to STDOUT in order to send to Asterisk, and should read from STDIN, in order to receive information back from Asterisk
- Commands sent to Asterisk must be terminated with a newline character
- The result returned by AGI commands is a text string, generally of the form: 200 Result=<number>. Some commands may return more information in addition to the Result code
- If you send Asterisk an invalid command your result will be 510 Invalid or unknown command
- All commands return a result. Commands which don't really need to return a result return 0 as the number

AGI program execution flow

- At script startup time Asterisk sends various pieces of information to your script and you should read these items, via standard input, before doing much else.
- Each item is sent on a line terminated with a newline and the end of the list is indicated by an empty line.
- The list of items received will look something like this:
  agi_request: test.py
  agi_channel: Zap/1-1
  agi_language: en
  agi_type: Zap
  agi_callerid:
  agi_dnid:
  agi_context: default
  agi_extension: 3
  agi_priority: 1
- If you need the information provided then save it; otherwise feel free to throw it away.

Things to remember about AGI programming

- Upon execution, Asterisk will fork an asterisk child process to run the AGI.
- Each AGI process being run takes its toll on the system.
- Executing the “Dial” Asterisk application will halt the currently running AGI program, till the Dial application ends.
- Executing the “Dial” Asterisk application within an AGI application will leave the AGI process hanging in the user space, taking its toll on the operating system resources and user space resources.
- An AGI program is identical to a CGI-BIN interface program, with a slight difference in the user I/O stream operation. If your AGI input methods do not cover all aspects of the user’s input, security issues may arise or unexpected system responses.

AGI program example: In C

```
#include <stdio.h>

main() {
   char line[80];
   /* use line buffering */
   setlinebuf(stdout);
   setlinebuf(stderr);
   /* read and ignore AGI environment */
   while (1) {
        fgets(line,80,stdin);
        if (strlen(line) <= 1) break;
   }
   /* Send asterisk a command */
   printf("SAY NUMBER 123 \"\"\n");
   /* Read response from Asterisk and show on console */
   fgets(line,80,stdin);
   fputs(line,stderr);
}
```

Asterisk::AGI – A PERL Module for Asterisk AGI programming

- James Golovich had developed a complete library of functions to be utilized with PERL.
- For more information about using PERL for AGI programming, go to:

  http://asterisk.gnuinter.net/

AGI programming : Ala PHP Style

```
#!/usr/bin/php4 -q
        <?php
        ob_implicit_flush(true);
        set_time_limit(6);
        $in = fopen("php://stdin","r");
        $stdlog = fopen("/var/log/asterisk/my_agi.log", "w");

        // toggle debugging output (more verbose)
        $debug = false;

        // Do function definitions before we start the main loop
        function read() {
          global $in, $debug, $stdlog;
          $input = str_replace("\n", "", fgets($in, 4096));
          if ($debug) fputs($stdlog, "read: $input\n");
          return $input;
        }

        function errlog($line) {
          global $err;
          echo "VERBOSE \"$line\"\n";
        }

        function write($line) {
          global $debug, $stdlog;
          if ($debug) fputs($stdlog, "write: $line\n");
          echo $line."\n";
        }
```

AGI programming : Ala PHP Style (cont…)

```
// parse agi headers into array
while ($env=read()) {
  $s = split(": ",$env);
  $agi[str_replace("agi_","",$s[0])] = trim($s[1]);
  if (($env == "") || ($env == "\n")) {
    break;
  }
}

// main program
echo "VERBOSE \"Here we go!\" 2\n";
read();
errlog("Call from ".$agi['channel']." - Calling phone");
read();
write("SAY DIGITS 22 X");
read();
write("SAY NUMBER 2233 X");
read();

// clean up file handlers etc.
fclose($in);
fclose($stdlog);

exit;
?>
```

AGI Command Reference

ANSWER
AUTOHANGUP <time>
CHANNEL STATUS [<channelname>]
EXEC <application> <options>
GET DATA <filename> [<timeout>] [<max digits>]
GET VARIABLE <variablename>
HANGUP [<channelname>]
RECEIVE CHAR <timeout>
RECORD FILE <filename> <format> <escape digits> <timeout> [BEEP]
SAY DIGITS <digit string> <escape digits>
SAY NUMBER <number> <escape digits>
SEND IMAGE <image>
SEND TEXT "<text to send>"
SET CALLERID <number>
SET CONTEXT <desired context>
SET EXTENSION <new extension>
SET PRIORITY <new priority number>
SET VARIABLE <variablename> <value>
STREAM FILE <filename> <escape digits>
TDD MODE <on|off>
VERBOSE <level>
WAIT FOR DIGIT <timeout>

AGI Execution upon call hang-up: DeadAGI execution

- It is possible to run an AGI program upon the hang-up of a channel, via the “DeadAGI” program.
- In order for the dial-plan to run an AGI upon a channels hang-up, the context accepting the call should have the hang-up extension defined, with a DeadAGI execution path.
- Example:

  exten => h,1,deadagi,test.php
- One of the issues of using DeadAGI is the fact that all variables that had been used in the previous AGI invocation, are no longer available. Preserving the state and affinity of the AGI program should be done externally, via a persistent storage (File or Database).

Sources of information about Asterisk and AGI Programming

- http://www.asterisk.org – The Asterisk Project Web Site
- http://www.voip-info.org – An all VoIP information site with extensive Asterisk resources.
- http://www.asterisk.org.il – The Israeli chapter of the Asterisk project (not yet fully available).
- http://www.asterisk.co.il – The Israeli Asterisk shop, to purchase Digium and Asterisk related hardware online (not yet fully available).
- http://www.dimitel.com – The Dimi Telecom website, with technical forums where you can pick our brains.
- http://asterisk.gnuinter.net - James Golovich's PERL AGI modules.

Thank you very much for listening

Questions Anyone?