VoIP Workshop

PacNOG 5

Papeete, French Polynesia

June 2009

Labs 1 - 4, Asterisk

Lab 5, INOC-DBA

Lab 6-7, Cisco Voice Gateways

Lab 8, CODECS

Lab 9, SIP Call Flow Analysis

Lab 10, Syslog and Call Detail Records (CDRs)

Lab Summary

Server logins: user inst, password pacn0g2k9

Servers are statically assigned IPs from, as follows:

Group #	Extensions	Server IP	Gateway	FXS Port	FXO Port
01	0100 - 0199	192.168.2.101	c2610xm-pots1	1	1
02	0200 - 0299	192.168.2.102	c2610xm-pots1	1	1
03	0300 - 0399	192.168.2.103	c2610xm-pots1	1	1
04	0400 - 0499	192.168.2.104	c2610xm-pots1	2	2
05	0500 - 0599	192.168.2.105	c2610xm-pots1	2	2
06	0600 - 0699	192.168.2.106	c2610xm-pots2	1	1
07	0700 - 0799	192.168.2.107	c2610xm-pots2	1	1
08	0800 - 0899	192.168.2.201	c2610xm-pots2	2	2
09	0900 - 0999	192.168.2.202	c2610xm-pots2	2	2
10	1000 - 1099	192.168.2.203	c2610xm-pots3	1	1
11	1100 - 3199	192.168.2.204	c2610xm-pots3	1	1
12	1200 - 3299	192.168.2.205	c2610xm-pots3	1	1
13	1300 - 3399	192.168.2.206	c2610xm-pots3	2	2
14	1400 - 3499	192.168.2.207	c2610xm-pots3	2	2

c2610xm-pots-1	192.168.1.151
c2610xm-pots-2	192.168.1.152
c2610xm-pots-3	192.168.1.153
c2610xm-e1-1	192.168.1.154
c2610xm-e1-2	192.168.2.254
as5400-e1-1	192.168.2.126

Login details for Cisco boxes (SSH only!): user:voip, password:discovoice

Dial Plan for each group:

'**??**' is your group number, and provides the first two digits of your extension numbers.

??00 - **??**09	SIP phones (password for each, *extension+*passwd, e.g. 2000passwd)
??10 - **??**19	Fun with IVRs
??20 - **??**29	Music on hold extensions
??30	DB count application
??40 - **??**49	Agents
??50	'Helpdesk' queue access
??60	Echo test
??70	Conference bridge
??80 - **??**89	Festival text to speech play extensions
??90	Ring the VG200 connected analog phone
??99	Voicemail access

9. Access an outside PSTN line
8. Send calls out via an e1 line (c2600xm-e1-x)
7. Send calls out via a POTS analog line (c2600xm-pots-x)
6. Access INOC-DBA
5. Send calls to another group

Lab 1: Initial Asterisk Install

You may need to tell your Ubuntu box to NOT look for it's CD. Goto System -> Administration -> Software Sources. Uncheck the box 'CDRom with Ubuntu x.xx' then click close.

1. Install Asterisk

```
aptitude install asterisk
aptitude install asterisk-sounds-extra
(for ubuntu 7.10 desktop, you'll need to run the following:)
  aptitude install libsnmp-dev
apt-get build-dep asterisk
```

We need to download and build the latest zaptel source code so as to get the ztdummy module. This is required as a timing source for applications such as meetme conferencing and music on hold.

Download and untar the Zaptel source code:

```
wget http://downloads.digium.com/pub/zaptel/zaptel-1.4-current.tar.gz
tar xzvf zaptel-1.4-current.tar.gz
```

Compile the Zaptel module (replace xx with the correct zaptel version):

```
cd zaptel-1.4.xx

make clean
./configure
make
make install
make config
```

load ztdummy module:

```
modprobe ztdummy
```

check that zaptel and the ztdummy driver are loaded

```
lsmod | grep zaptel
```

You should see something that looks like the following:

```
jonny@vegas:~$ lsmod | grep zaptel
zaptel                193284  1 ztdummy
crc_ccitt               3072  1 zaptel
```

Set RUNASTERISK=yes in /etc/default/asterisk

```
nano /etc/default/asterisk
```

2. Start Asterisk

```
/etc/init.d/asterisk start
```

Have a look at the available startup options:

```
asterisk -h
```

To connect to the Asterisk CLI:

```
asterisk -r
```

3. Edit Configuration Files in /etc/asterisk/

Set up three SIP peers in sip.conf: **??**00, **??**01, **??**02 (where '**??**' are the extension numbers assigned to your group). Add to the bottom of sip.conf, repeating for each of the three SIP peers:

```
[??00]
type=friend
host=dynamic
username=??00
secret=passwd??00
canreinvite=no
nat=yes
context=phones
dtmfmode=rfc2833
allow=all
```

Create backup of original extensions.conf:

```
mv extensions.conf orig_extensions.conf
```

Create new extensions.conf with the following:

```
[general]
static=yes
writeprotect=no
autofallthrough=yes
clearglobalvars=no
priorityjumping=yes

[phones]
; remember to replace ?? with your group's numbers!
exten => ??00,1,Dial(SIP/??00)
exten => ??01,1,Dial(SIP/??01)
exten => ??02,1,Dial(SIP/??02)

exten => ??60,1,Answer()
exten => ??60,2,Playback(demo-echotest)
exten => ??60,3,Echo
exten => ??60,4,Playback(demo-echodone)
exten => ??60,5,Hangup
```

Connect to Asterisk (asterisk -r), up the debug output (set verbose 10), and reload the config (reload).

4. Configure Softphone

Download and configure the Xten Xlite Softphone - (http://www.xten.com/index.php?menu=download)

Input SIP settings in *Main Menu > System Settings > SIP Pro.. > Default*

Enabled:	Yes
Username:	SIP extension you are configuring (e.g. 2000)
Authorization User:	Same as *Username*
Password:	extensionpasswd, e.g. 2000passwd

SIP Proxy:	The address of your Asterisk server
OutBound Proxy:	Same as *SIP Proxy*

You should now be able to call between your three phones.

Call the echo test on **??**60 and you should be able to hear yourself!

Lab 2: Basic Asterisk Config

Configure the following, using the extensions given in the Lab summary:

- voicemail for each extension
- a sample IVR
- a meetme conference
- a sample MOH stream

Here's a start on the configuration files:

voicemail.conf

```
[default]
??00 => 1234,User 1,user1@email.address
??01 => 1234,User 2,user2@email.address
??02 => 1234,User 3,user3@email.address
```

extensions.conf

```
[phones]
; configure pattern match for local extensions
; e.g. _200X
exten => _??0X,1,Dial(SIP/${EXTEN},15)
exten => _??0X,n,Voicemail(u${EXTEN})
exten => _??0X,n,Hangup()

; allow checking of voicemails. try it out!
exten => ??99,1,VoicemailMain()

; extension to allow dialling the IVR
exten => ??10,1,Goto(ivr-test,s,1)

[ivr-test]
; based on the slides, create an IVR which allows you to
; ring your extensions
```

If you're not sure about how specific applications work, from the Asterisk CLI try:

```
show applications
show application goto
```

Lab 3: Advanced Asterisk Configuration

1. Asterisk Database

Implement the following in extensions.conf:

```
[phones]
; start counting and store count progress in astdb
```

```
; check if DB key exists, if not, jump to key_no_exist
; function DB_Exists returns 1 if the key exists, 0 if not
exten => ??30,1,GotoIf(${DB_EXISTS(test/count)}?:key_no_exist)

; begin the counting!
exten => ??30,n(start),Set(COUNT=${DB(test/count)})
exten => ??30,n,SayNumber(${COUNT})
exten => ??30,n,Set(COUNT=$[${COUNT} + 1])
; update the DB
exten => ??30,n,Set(DB(test/count)=${COUNT})
exten => ??30,n,Goto(start)

; if we got here it is because the key didn't exist in the DB
; create the key
exten => ??30,n(key_no_exist),Set(DB(test/count)=1)
; and jump back to the start to begin counting
exten => ??30,n,Goto(start)
```

Reload Asterisk, and have a look at the Asterisk DB

```
reload
database show
```

Now dial ..30, and look at the DB again. You should see a new key (test/count) in the DB containing the current count.

2. Implement Nightmode

We want the nightmode to work as follows:

We want to create an extension called ..50 for our 'main number'
We will create two new keys in the DB:
nightmode/open_time, and nightmode/close_time
When a call comes in, we will check to see if we are currently between those two times, and if so ring all three phones. If not, go straight to voicemail

Hints:

To manually set a DB key from the CLI:

```
database put family key value
```

Time based branching:

```
show application gotoif
```

Dialling multiple channels simultaneously:

```
Dial(SIP/1000&SIP/2000&SIP/3000)
```

3. Extension Macro

Look in the original /etc/extensions.conf (you should have moved it to orig_extensions.conf), and use it as a guide.

Create a simple extension macro to dial our extensions and branch to voicemail if not answered.

4. Set up Agents

Edit agents.conf - add three agents for you group to the bottom of the existing file:

```
agent => ??40,1234,Agent one
agent => ??41,1234,Agent two
agent => ??42,1234,Agent three
```

To enable Agent login and logout, add to extensions.conf:

```
[phones]
; hint in CLI, show application AgentCallbackLogin
exten => ??59,1,AgentCallbackLogin()
```

Reload Asterisk, then check the state of Agents before and after a login:

```
show agents
```

5. Set up a Queue

Edit queues.conf - use the existing defaults as a guide. Call the queue helpdesk (this is at the start of the file in [ ]). The important piece is to add to the bottom of queues.conf:

```
member => Agent/??40
member => Agent/??41
member => Agent/??42
```

And in Extensions.conf create a means to enter the queue:

```
[phones]
exten => ??50,1,Queue(helpdesk)
```

Ring the queue with Agents all logged out, and all logged in.

6. Install Festival text to speech

Exit out of Asterisk and install Festival:

```
apt-get install festival
```

Configure Festival for Debian / Ubuntu. Make /etc/festival.conf look like the following:

```
;; Enable access to localhost (needed by debian users)
(set! server_access_list '("localhost\\.localdomain" "localhost"))

;; set italian voice (comment the following 2 lines to use british_ameri-
can)
(language_italian)
(set! voice_default 'voice_pc_diphone)

;;; Command for Asterisk begin
(define (tts_textasterisk string mode)
    "(tts_textasterisk STRING MODE)
  Apply tts to STRING.  This function is specifically designed for
  use in server mode so a single function call may synthesize the string.
  This function name may be added to the server safe functions."
      (utt.send.wave.client (utt.wave.resample (utt.wave.rescale (utt.synth
                  (eval (list 'Utterance 'Text string))) 5) 8000)))
;;; Command for Asterisk end
```

To use Festival:

```
exten => 123,1,Festival('Hello World')
exten => 123,2,SetVar(speech='Hello World by variable')
exten => 123,3,Festival('${speech}')
```

Lab 4: Asterisk Exercises

1. Another Extensions Macro
Write an extension macro which looks up a database to get the following information:

callerID name
callerID number
Voicemail box
do not disturb flag

If the do not disturb flag is set, playback a prompt saying (sorry, <name> doesn't want to be disturbed). Make sure the macro correctly set the CallerID name and number.

2. DB lookup for incoming calls
Write a piece of code that does a DB lookup on inbound calls into the [incoming] context, looks up the number in the database, and uses the result to branch into the appropriate location in the dial plan.

In what circumstances do you think this would be handy?

3. Write a Prompt recording Macro.
This macro will need to take as input the filename to record, and optionally the format to record it in.

The macro needs to:

1. record the prompt
2. let the user play it back
3. let the user confirm they wish to use that prompt
4. save the prompt in the correct location

Note, Festival text to speech is handy to provide instructions here!

4. Write an application to ping a device
Create a context (starting with the 's' extension) which allows you to ping a device.

You'll need to work out how to accept DTMF input, run a ping command external to asterisk, and read the result back to the caller.

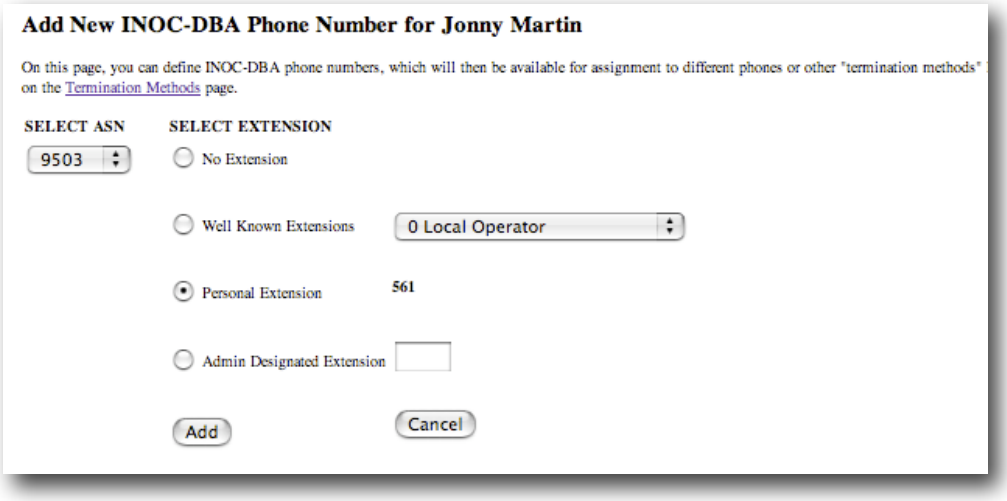
Lab 5: Connecting Asterisk to INOC-DBA

You will need to have set up an account and log in to the INOC-DBA administration system to do this.

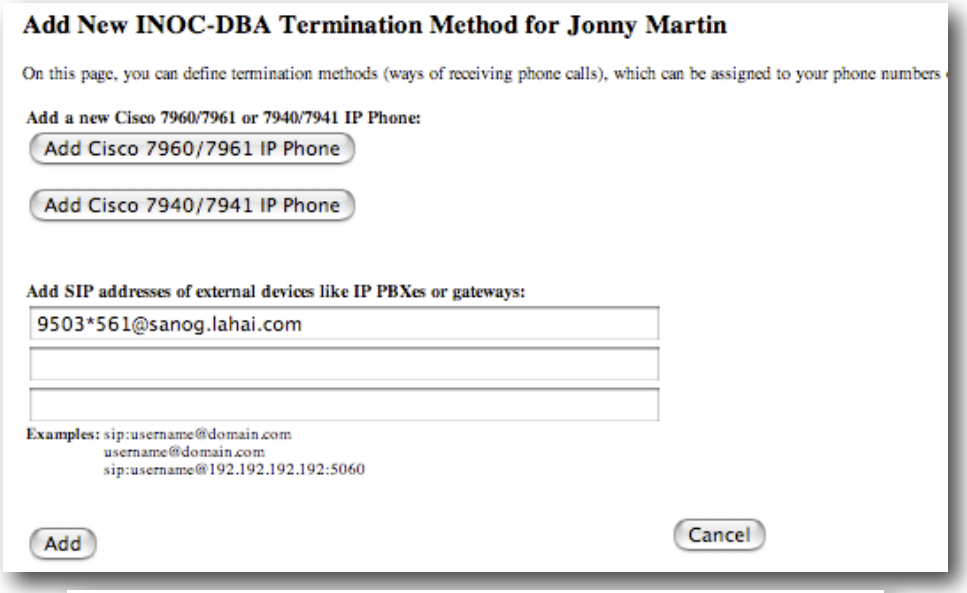
http://www.pch.net/inoc-dba/

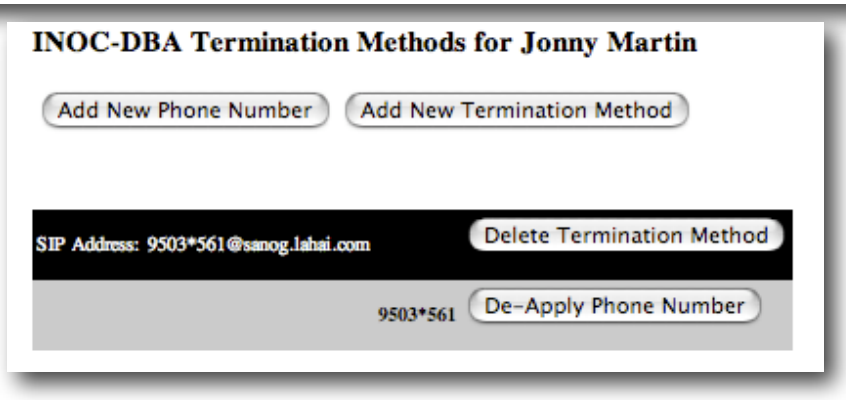
1. Set up INOC-DBA to send calls to your Asterisk server.
You need to set up a termination method through the INOC-DBA system to deliver calls to your asterisk server.

For this lab exercise we will will set up your INOC-DBA personal extension to terminate calls on your lab asterisk server. Select 'My Phone Numbers' from the menu and populate it accordingly:



Select 'Termination Methods' from the menu and add the IP address of your lab server, then select that as the termination method for your personal extension.





2. Configure Asterisk sip.conf

Asterisk needs to be configured to SIP REGISTER itself with the INOC-DBA servers. Add the following to the [general] section of sip.conf:

```
;need to add the register line, which is what Asterisk
;will send to the inoc-dba. the format is
;
; register = > ASN*EXT:password:username@inoc-dba.pch.net/Local_extension
;
```

```
register => 9503*561:password:jonny@inoc-dba.pch.net/9503*561
```

Replacing 9503*561 with your INOC-DBA extension, and password:jonny@ with your password and login name.

This statement registers our Asterisk box with INOC-DBA. Inbound calls are sent to the default context.

3. Configure inbound calls
Inbound calls land in the default context. We want these calls to ring a phone, so add something like the following into the [default] context, substituting your details for SIP/2000 and 9503*561.

```
exten => 9503*561,1,Dial(SIP/2000,15)
exten => 9503*561,n,Voicemail(u2000)
exten => 9503*561,n,Hangup()
```

You may want to have inbound NOC calls ring multiple phones. Configure your INOC-DBA extension to ring multiple phones at once.

A nicer way to implement this is to use a GoTo statement in the default context to send inbound calls to 9503*561 to an extension elsewhere in your dialplan, enabling you to easily change the destination for inbound calls. Use this method to send a call to one of your existing extensions. This could be a phone, voicemail, conference, etc.

4. Configure outbound calls
First set up a new SIP peer for INOC-DBA. Add the following peer to sip.conf:

```
[inoc-dba]
type=friend
host=inoc-dba.pch.net
port=5060
username=pchconf
fromuser=9503*561
secret=nothing
canreinvite=yes
context=from-inoc-dba
insecure=very
```

(Remember to use the correct ASN*ext for the fromuser line in the sip peer)

Calls prefixed with a 9 will be sent out to INOC-DBA. We need to first strip the 9, and then set our outgoing callerID correctly. Add the following to the appropriate context in extensions.conf:

```
; This extension is for outgoing calls to inoc-dba
; 9 for an outside-inoc-dba-line
exten => _9.,1,Set(CALLERID(all)=Jonny Martin <9503*561>)
exten => _9.,n,Dial(SIP/${EXTEN:1}@inoc-dba)
exten => _9.,n,Hangup
```

Lab 6. Cisco IOS Voice Gateway i.

In this lab we are going to setup a Cisco router with one FXS interface such that an attached telephone can dial all of the allocated extensions on each asterisk server.

1. Configure SIP peer on Asterisk server
You need to configure a SIP peer for both in and outbound calls to the cisco gateway. In sip.conf add the following peer definition:

```
[cisco]
type=friend
disallow=all
allow=ulaw
canreinvite=no
context=from-cisco
host=a.b.c.d   ; see instructor for IP address to use
dtmfmode=rfc2833
insecure=very
```

Note, if you don't have a context [from-cisco] in extensions.conf you will need to create it, and put some extension statements in it!

2. Configure extensions.conf
Inbound calls will land in the [from-cisco] context. Use a GoTo statement to send these calls to the context where you have all your useful extensions defined.

Next we want to create an extension that when dialled from our Asterisk system will dial the cisco analog phone. Add the following the appropriate context to extensions.conf. This example is for group one, using extension 2190

```
exten => 2190,1,Dial(SIP/2190@vg200)
```

The IOS gateway is simply another SIP peer, so you can send calls to it in the same fashion you send calls to any other peer.

3. Configure the VG200
The FXS voice port is already configured for you. In this case, the default IOS settings are all that is required.

Create a dial-peer to ring the analog phone. Login to the gateway and configure the following (changing 2190 for your group's extension, e.g. 2190, 2290):

```
dial-peer voice 2190 pots
 destination-pattern 2190
 port 1/0/0
```

(Remember to replace 21 2190 with your group's allocated extension digits)

Create a dial-peer to send calls to your asterisk server (again, an example for group 1):

```
dial-peer voice 1 voip
 description calls to group 1
 destination-pattern 21..
 session protocol sipv2
 session target ipv4:a.b.c.d  ! insert your ip address here
 dtmf-relay rtp-nte
 codec g711ulaw
 no vad
```

(Note here that the '..' in the Cisco config is supposed to stay as '..' - a '.' matches exactly one digit in Cisco pattern match terminology.

Test that you can make calls to and from the analog phone.

Lab 7. Cisco IOS Voice Gateway ii.

In this lab we are going to setup an E1/T1 interfaces on a Cisco router as an outbound PSTN gateway.

In this example, calls from your softphone to the asterisk server will be prefixed with a 7. Asterisk will send these calls to the Cisco gateway without stripping the 7.

In our fictitious example, calls will be going out to the local PSTN and requires to be prefixed by a 0. The router will first strip the 7, and append a 'telco' access code of 0 before sending the call out one of the E1/T1 interfaces.

The digit stripping is performed on the inbound dial-peer (VoIP in this case), and the prepend on the outbound dial-peer (POTS in this case).

1. Configure SIP peer on Asterisk server
You need to configure a SIP peer for both in and outbound calls to the AS5400 gateway. In sip.conf add the following peer definition:

```
[cisco-e1]
type=friend
disallow=all
allow=ulaw
canreinvite=no
context=from-cisco-e1
host=a.b.c.d  ; check with instructor for address
dtmfmode=rfc2833
insecure=very
```

Note, if you don't have a context [from-cisco-e1] in extensions.conf you will need to create it, and put some extension statements in it!

2. Configure extensions.conf
Inbound calls will land in the [from-cisco-e1] context. Use a GoTo statement to send these calls to the context where you have all your useful extensions defined. In this case we want inbound calls to ANY number to be met with music on hold.

We want to create an extension match that matches strings starting with a 7, and send the call to the router:

```
exten => _7.,1,Dial(SIP/${EXTEN}@as5400)
```

Why do we use ${EXTEN} in the dial string in this case?

3. Configure the E1 gateway
The E1/T1 port will already be configured for you. First we need to create a translation rule to strip the leading 7 of incoming VoIP calls. This translation rule will be called 10+<group number> e.g. 101 for group one.

```
translation-rule 101
 Rule 1 ^71.% 1
 Rule 2 ^72.% 2
 Rule 3 ^73.% 3
 Rule 4 ^74.% 4
 Rule 5 ^75.% 5
 Rule 6 ^76.% 6
 Rule 7 ^77.% 7
 Rule 8 ^78.% 8
 Rule 9 ^79.% 9
```

The translation rule for prepending a 0 for outbound POTS calls is a little simpler. This rule will be called 20+<group number> e.g. 201 for group one.

```
translation-rule 201
 Rule 1 ^.% 0
```

Create a dial-peer to match incoming VoIP calls from your asterisk server (remember that a call consists of two dial-peers. This dial-peer will be called 100+<group number>, e.g. 1001 for group one:

```
dial-peer voice 1001 voip
```

```
 description calls from group 1
 answer-address 21..
 destination-pattern 21..
 translate-outgoing called 101
 session protocol sipv2
 session target ipv4:a.b.c.d  ! see instructor for ip address
 dtmf-relay rtp-nte
 codec g711ulaw
 no vad
```

The answer-address line tells this dial-peer to match incoming calls from only your asterisk server. In this case, session protocol and target is correctly setup, however at this stage it is not being used as no inbound calls are configured.

Create a dial-peer to send calls out the first E1/T1 interface on the router. This dial-peer will be called <group number>, so substitute your group's number when configuring this. We will be using the the translation pattern to prepend a 0 that we configured up earlier. For group one:

```
dial-peer voice 1 pots
 destination-pattern T
 translate-outgoing calling 201
 direct-inward-dial
 port 1/0:D
```

Once you have completed this, try making a call. If it fails, perform some debug both on your Asterisk server, and on the gateway to try and work out why.

Some useful IOS voice commands:

```
show call active voice brief     ! show voice calls currently in progress
show call history voice brief    ! show recent voice calls

show controller controller-id    ! show details of interface controller
show voice port                  ! show voice port configuration

show dial-peer voice [summary | id#]

test translation-rule            ! test a translation pattern
     e.g. test translation-rule 201 64212304323
```

Lab 8. CODECs

This lab aims to give you real world experience making calls over real satellite IP links with different CODECs and settings.

We will be making calls to an asterisk server back in New Zealand to an echo test, and the PSTN.

1. Asterisk configuration
You'll need the following SIP peer configured on your asterisk server (use apricot<group number> for the peer name):

```
[nz]
type=friend
disallow=all
allow=alaw
callerid=your name <2100>
dtmfmode=rfc2833
canreinvite=no
nat=no
```

```
host=a.b.c.d
username=voip1
fromuser=voip1
secret=voip1
```

We'll se our dialplan up such that calls starting with 1 are sent to NZ, after having the 1 stripped:

In extensions.conf add a route to the NZ asterisk box:

```
exten => _1.,1,Dial(SIP/${EXTEN:1}@nz)
```

2. Check connectivity to NZ server
We want to see what the connectivity to the NZ server looks like, a ping and traceroute will give us an idea of what this is like. From a shell prompt on your asterisk server:

```
ping a.b.c.d
mtr a.b.c.d
```

What does the path look like?

What is the round trip latency? What might a voice call sound like?

3. Make some calls
Make a call to the NZ server. The NZ server is configured with the following dialplan:

200	echo test
222	music on hold
0...	domestic NZ numbers (e.g. 021 2304323 to dial Jonny's cellphone)
00...	international numbers (e.g. 00 682 xxxxx to dial Rarotonga)

(Careful when making PSTN calls, as call quality will be highly variable due to the more limited bandwidth into NZ, and the number of simultaneous call attempts!)

Once you have successfully made a call, try further calls to the echo test and music on hold using other co-decs. Change the 'allow=' line in your nz SIP peer to try the following codecs:

gsm	- GSM codec
ilbc	- Internet low bandwidth codec
ulaw	- G.711 ulaw
speex	- Speex codec
g726	- G.726.1 codec

Lab 9. SIP Call Flow Analysis

This lab will cover the installation of the sip_scenario call flow analysis tool. This provides a pretty html interface to captured SIP call traffic.

1. Install SIP Scenario Callflow Generator

```
cd /home/voip/
mkdir callflow
cd callflow
wget http://www.iptel.org/~sipsc/index/sip_scenario.v1.2.7.zip
unzip sip_scenario.v1.2.7.zip
```

edit sip_scenario.pl and change the first line from:

```
#!/usr/local/bin/perl -w
```

--to--

```
#!/usr/bin/perl -w
```

2. Capture sip traffic:

To capture all traffic (careful, there might be a lot of traffic!):

```
tcpdump -s0 -w capture_filename
```

To capture just sip traffic:

```
tcpdump -s0 -w capture_filename2 port 5060
```

3. Generate pretty callflow diagrams

```
./sip_scenario.pl capture_filename
```

Then drop the html files in a webserver directory, or open them directly with the file browser on your Linux machine.

Lab 10. Syslog and Call Detail Records

In this lab we are going to set up a syslog server on or lab servers, and configure our gateways to send Call Detail records to it.

1. Server Side syslog install

Syslog is quite limited, so use syslog-ng

```
aptitude install syslog-ng
```

This will remove the old syslog package and replace it with syslog-ng which is configured to replicate how syslog was configured.

```
nano /etc/syslog-ng/syslog-ng.conf
```

Add;

```
source s_remote {
      udp();
};
```

This adds a new source called s_remote and accept syslog messages fromthe syslog UDP port.

```
destination df_remote{ file("/var/log/remote/$HOST.log");  };
```

This creates a new syslog logging target which is a file, it will use the $HOST macro which will be filled in with the source address the messages came from. If in the options section use_dns(); is set to no then it will use just the IP address, or if it is set to yes it will use the RDNS for the IP.

Now we create a bridge between the logging source and destination like this;

```
log {
   source(s_remote);
   destination(df_remote);
};
```

restart syslog-ng, create the /var/log/remote/ directory and you are away. This is not the most secure method of setting it up, but the fastest, if it is going to be used in the real world you should also set up filters to ensure that not just anyone can log to you and also to perhaps use filequotas so that it doesn't fill your disks up. Also you should set up logfile rotations.

2. Configure Cisco gateway to send syslog messages

On the Cisco box configuration is pretty simple.

```
logging host A.B.C.D
logging trap 0-7 (0 being almost nothing, 7 being the most verbose)
```

If you Cisco supports it you might want to log your call records via syslog as well. The AS5400 supports this but the c2600XMs don't.

```
calltracker enable
calltracker call-record verbose
```

Lab 11. Cisco QoS

In this lab we are going to set up very basic QoS between two Cisco routers to provide priority to voice traffic

1. Configure Access list for marking

Note that we can only configure one access list so make sure you add to the following access list on the router, and don't replace what is already there!

This needs to be done on the router that has your analogue phone connected to it.

```
access-list 100 remark ACL to match VOIP traffic
access-list 100 permit udp any any eq 4569
access-list 100 permit udp any any eq 5004
access-list 100 permit udp any any eq 5060
access-list 100 permit ip any host INSERT-YOUR-SERVER-IP-HERE
```

We will use this access-list to match voice media and signalling traffic.

What traffic is the explicit ports matching?

2. Configure Low Latency Queuing

Configuring LLQ for voice traffic on cisco routers is quite straight forward. Configure the following on your router ('!' represents a commented line, there is no need to type those lines into the router).

```
! create a class that matches the traffic in the access-list above
class-map voip
 match access-group 100
!
! create a policy map to queue based on our class called 'voip'
policy-map llq-voip
 class voice
  ! configure 160kbps priority bandwidth for class voip
  priority 160
```

```
 class class-default
  fair-queue
```

3. Apply the service policy to an interface

We need to apply our service-policy to an interface now. In our case we are using the onboard fast ethernet interface to send traffic to our gateway.

```
interface fa0/0
 service-policy output llq-voip
```

In which direction has this QoS configuration affected out traffic?

How might we handle inbound traffic?

4. Verifying QoS operation

Verfying QoS from the router CLI is achieved with the following commands:

```
show policy-map <policy-map-name>
show class-map <class-map-name>
show policy-map interface <interface>
show interface <interface>
```

The Show Policy-map command displays the contents of a policy map, including the priority setting in a specific policy map if one is included. It shows the classes that belong to the priority queue along with what traffic belongs to the default-class. It should be noted the default queue uses weighted-fair queuing.

The Show class-map command shows the criteria for a packet to become a member of a class.

The Show policy interface command shows what class-maps are configured under a policy-map applied to the named interface. You can determine if the desired QoS policy is active on the interface, and how much of the traffic meets the requirements to become a member of the class. It also tells how many packets have been dropped from the output queue. If there are drops present it is because congestion was present in the circuit and the queues started to drop packets to make way for the high priority traffic.

The show interface command gives you the interface statistics, and it also gives you how many output queue drops have occurred on the interface. This is a quick way to determine if there has been congestion and if packets are being dropped. It also tells you if the QoS commands are applied to the physical or logical interface. In this case, it is applied to the logical because the queuing mechanism is for each virtual channel or VC.

5. Hierarchal Shaping

Implement a hierarchal shaper based on the following configuration. By using a very low shaping target for the parent policy-map you will be able to easily exercise the child shaper.

```
policy-map shaper-parent
 class class-default
  shape average 80000
  service-policy shaper-child

policy-map shaper-child
 class voip
  priority 80
```

Other Lab Notes

1. Meddling with packets

In the lab we used the tc (traffic control) linux command to modify packets. This allowed us to set drop, latency, and jitter probabilities to enable us to simulate the parameters of real links. We could also take things to the extreme and test how various VoIP codecs and devices perform across very bad links.

Full documentation for the use of tc is at: http://www.linuxfoundation.org/en/Net:Netem

2. Connecting two Asterisk boxes together via a SIP trunk.

Example config, connecting group 1 to group 2 via an authenticated SIP trunk:

GROUP 1

sip.conf

```
[group2]
type=friend
host=169.223.129.202
allow=all
username=group1
fromuser=group1
secret=supersecret
nat=no
context=from-other-groups
dtmfmode=rfc2833
```

extensions.conf

```
[phones]
exten => _02XX,1,Dial(SIP/${EXTEN}@group2)

[from-other-groups]
exten => _01XX,1,GoTo(phones,${EXTEN},1)
```

GROUP 2

sip.conf

```
[group1]
type=friend
host=169.223.129.201
allow=all
username=group2
fromuser=group2
secret=supersecret
nat=no
context=from-other-groups
dtmfmode=rfc2833
```

extensions.conf

```
[phones]
exten => _01XX,1,Dial(SIP/${EXTEN}@group1)

[from-other-groups]
```

```
exten => _02XX,1,GoTo(phones,${EXTEN},1)
```

3. tcpdump example

To capture packets from your softphone to your asterisk server, while filtering out ssh traffic:

```
tcpdump -i eth0 -n host 169.223.x.x and not port ssh
  (where x.x = ip of latop with softphone)
```