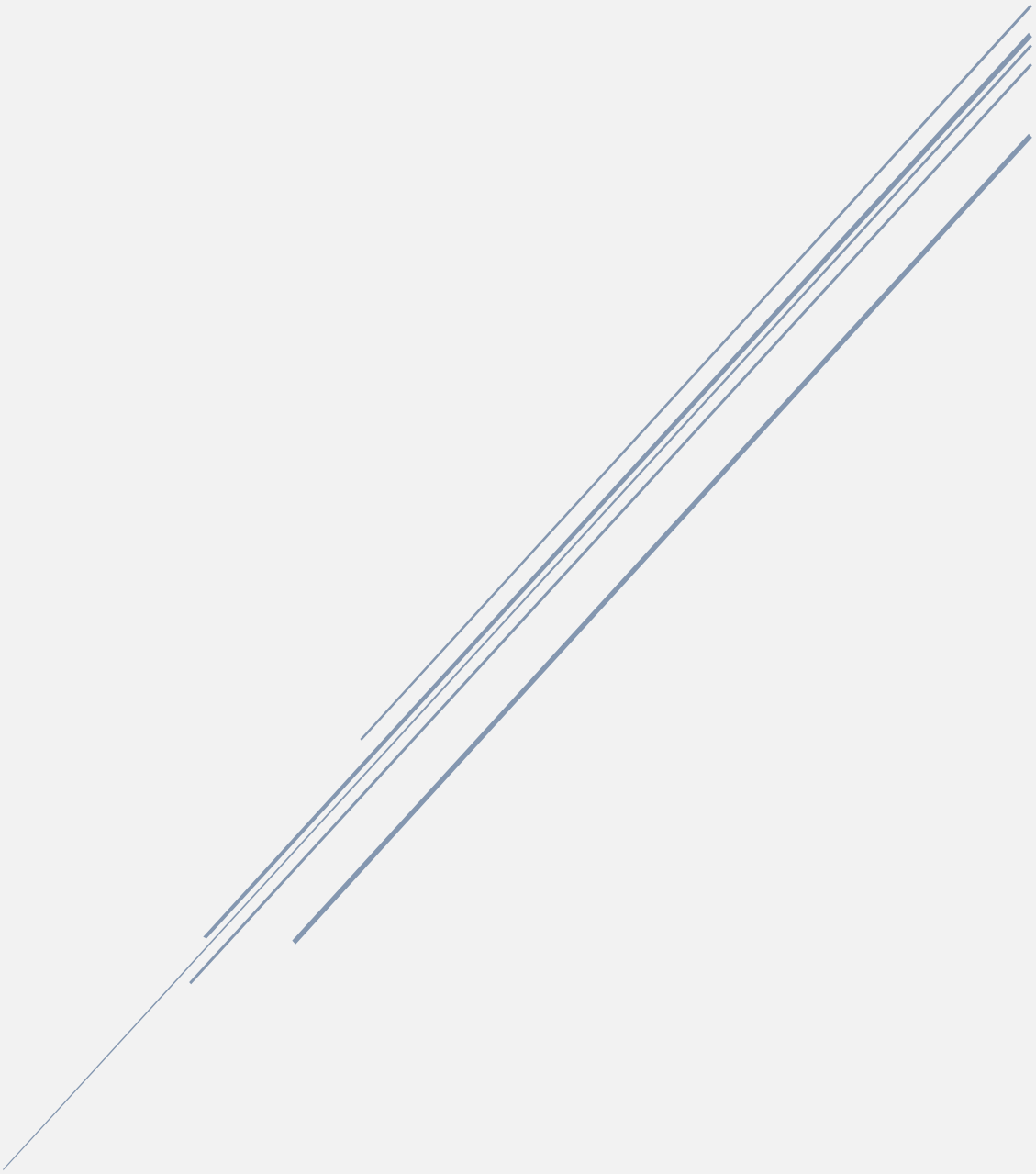
PRÁCTICA 2: CONFIGURACIÓN DE SISTEMAS DE TELEFONÍA IP

Laboratorio de Redes, Sistemas y Servicios



Silviu Constantin Sofrone

índice

Parte Inicial

Configuración inicial

Empezaremos editando el archivo *sip.conf*, que sirve para configurar todo lo relacionado con el protocolo de señalización SIP y añadir usuarios y terminales. En la sección **[general]** modificaremos algunas entradas referentes a la configuración por defecto de los usuarios y peers:

```
qualify=yes                    ; Permite monitorear la conecxion en los telfs VoIP
language=es                    ; Idioma español para todos los usuarios
disallow=all                   ; Desactivar todos los codificadores
allow=alaw, ulaw               ; Permitir codificadores en orden de preferencia
```

Por orden de preferencia, **alaw** codificadores utilizados en Europa y **ulaw** en Estados Unidos. También modificaremos **udpbindaddr=0.0.0.0** para que escuche todo.

A continuación, crearemos una plantilla (**[usuario]**) que nos sirva para todos los usuarios que vamos a registrar en la empresa de aquí en adelante:

```
[usuario](!)            ; Plantilla con la config a utilizar
type=friend             ; El usuario con esta extensión puede enviar y recibir llamadas
host=dynamic            ; Cualquier equipo con cualquier ip se puede registrar como cliente
context=labredes        ; contexto predefinido -> ver extensions.conf
```

Para crear las diferentes extensiones utilizamos **[nombre de la extensión](plantilla)** y les otorgaremos un nombre de usuario y una contraseña asociada. En nuestro caso, para simplificar y agilizar las pruebas, utilizaremos la misma contraseña para todas las extensiones.

```
; Extensión 601
[silviu](usuario)
username=silviu
secret=7777
```

Buzón de correo

En el archivo **voicemail.conf** (en la sección **[default]**) creamos los buzones de correo para cada extensión, formados por: el identificador (no tiene porque ser el mismo que la extensión, lo ponemos así para que sea más fácil reconocerlo), la contraseña correspondiente al buzón y el correo electrónico.

```
[default]
; Note: The rest of the system must reference

601 => 7777,buzon1,silviusofrone7@gmail.com
602 => 7777,buzon2,silviusofrone7@gmail.com
603 => 7777,buzon3,silviusofrone7@gmail.com
701 => 7777,buzon4,silviusofrone7@gmail.com
```

En **extensions.conf** definimos el comportamiento que vana tener las llamadas en nuestra centralita. Al marcar la extensión de un usuario que posea buzón, le indicamos que espere 15 segundos. En caso de que no responda dejamos un mensaje en el buzón correspondiente mediante la función **VoiceMail()** y el identificador del buzón asociado a dicho usuario:

```
[labredes]
exten => 601,1,Dial(SIP/silviu,15,tT)
        same => n,Answer()
        same => n,VoiceMail(601)

exten => 602,1,Dial(SIP/pepe,15,tT)
        same => n,VoiceMail(602)
```

Cada buzón tendrá asociado un menú al que se puede acceder llamando a un número que elijamos e introduciendo la contraseña que indicamos previamente en el archivo **voicemail.conf**. Debemos ser conscientes de que, aunque en nuestro caso todos los buzones tienen la misma contraseña y, por tanto, cualquier usuario puede acceder al buzón de otro, en una situación real tendríamos contraseñas únicas para cada uno.

```
; Buzones
exten => _301,1,VoiceMailMain(601)
exten => _302,1,VoiceMailMain(602)
```

Colas (queue)

Crearemos grupos para cada departamento, en los que las llamadas entrantes serán situadas en una cola de espera. De esta forma, si dentro de la empresa alguien no quiere comunicarse con alguien en específico del departamento, sino con cualquier miembro, llamará a su grupo y en base a la estrategia de llamada le contestará algún miembro lo antes posible.

En **queues.conf** creamos una sección referente al grupo. Los miembros se añaden con *member* y la estrategia de llamadas con *strategy* (en este caso un round robin con memoria):

```
[equipo700]

strategy = rrmemory
member => SIP/usuario1
member => SIP/usuario2
```

Utilizamos la función **Queue()** (en vez de utilizar **Dial(SIP/miembro1&SIP/miembro2,50)**) para llamar al grupo correspondiente. Dejamos las

demás opciones en blanco excepto por el timeout que será de 50 segundos, en caso de no contestar nadie del grupo se cuelga la llamada:

```
exten => 700,1,NoOp(Equipo ext 700)
        same => n,Answer()
        same => n,Queue(equipo700,,,,50)
        same => n,Hangup
```

Música en espera

```
[music_espera]
mode=files
directory=music_espera
```

En **musiconhold.conf** creamos un contexto para otro grupo que tenemos creado. Este grupo tiene la estrategia *ringall* (por defecto), que llama a todos los miembros hasta que alguien conteste. Al hacer la llamada se buscará el archivo correspondiente a la música de espera en el directorio que le indiquemos mediante *directory*. La ruta que seguirá *directory* será: **/usr/share/asterisk/nombre-del-directorio**, por tanto, en nuestro caso buscará en **/usr/share/asterisk/music_espera**.

Por tanto, procedemos a crear el directorio y almacenar en él la música de espera que deseemos:

- En **/usr/share/asterisk** creamos el directorio con: **mkdir music_espera**
- Cambiamos a Asterisk como propietario del archivo: **chown asterisk music_espera**
- Copiamos en el directorio la música de espera que queramos del directorio **moh**: **cp ../moh/manolo_camp-morning_coffee.gsm .**
- Cambiamos a Asterisk como propietario de la música de espera: **chown asterisk manolo_camp-morning_coffee.gsm**

En la consola podemos comprobar la nueva clase creada con el contexto que hemos creado:

```
ubuntuserver*CLI> moh reload
    -- Reloading module 'res_musiconhold.so' (Music On Hold Resource)
ubuntuserver*CLI> moh show classes
Class: default
        Mode: files
        Directory: moh
Class: music_espera
        Mode: files
        Directory: music_espera
ubuntuserver*CLI>
```

Mediante **Set(CHANNEL(musicclass)=music_espera)** *"seteamos"* la música de espera que acabamos de configurar:

```
exten => 600,1,NoOp(Equipo ext 600)
;Queue(queuename[,options[,URL[,announceoverride[,timeout[,AGI[,macro[,gosub[,rule[,position]]]
]]]]]])
        same => n,Set(CHANNEL(musicclass)=music_espera)
        same => n,Answer() ;Si no se contesta entra música en espera
        same => n,Queue(equipo600,,,,50) ;Sin opciones, URL,etc..solo timeout
        ;same => n,Dial(SIP/silviu&SIP/pepe,50) ;espera 50 segundos
        same => n,Hangup
```

Menú IVR

Creamos un contexto referente a llamadas procedentes del exterior de la empresa, al no encontrarse en el contexto **[labredes]**, estos usuarios no podrán comunicarse con los usuarios de la empresa. Para ello implementaremos un menú IVR que permita dichas comunicaciones.

```
[exterior]
type=friend
host=dynamic
context=phones
```

De esta forma, dentro de los contextos **[labredes]** y **[phones]** definimos el plan de llamada si se llama al menú, que será saltar al contexto **[ivr-1departamento]** mediante la función **Goto()**. Utilizamos dicho contexto para el menú como contexto único ya que, posteriormente, implementaremos más menús con selecciones desde teclado, por tanto, necesitamos que en cada menú las extensiones 1,2,3, etc. estén libres.

```
; Menú ivr
exten => 6800,1,GoTo(ivr-1departamento,s,1) ;Las llamadas a la extensión 6800 de dentro del con
texto labredes se redirigen al menú IVR, al igual que la llamadas del exterior dirigidas a dich
a extensión
```

```
[phones]
exten => 6800,1,GoTo(ivr-1departamento,s,1)
```

En el menú le indicamos al usuario que pulse 1 o 2 en base al departamento con el que quiera comunicarse. Aprovechando que tenemos creados dos grupos con extensiones pertenecientes a la empresa, los utilizaremos para referirnos a los departamentos de la empresa. De esta forma, a la hora de comunicar al usuario con un departamento llamaremos al grupo en cuestión y haremos uso de las colas antes mencionadas en vistas que nos responda cualquier miembro del departamento.

También iremos incrementando una variable que cuente el número de veces que el usuario no ha tomado ninguna decisión y utilizaremos un bucle para volver al audio original, para que tenga otra oportunidad. El resto de funcionalidades están comentadas en la siguiente captura y otras serán explicadas más adelante.

```
[ivr-1departamento]

exten => s,1,NoOp(IVR 1)        ;No hay opción, simplemente muestra por la consola de comnados Asterisk lo expuesto
        same => n,Answer
        same => n,Set(a=0)
        same => n(loop),Background(ivr-departamentos)
        same => n,WaitExten(20) ;Esperamos 20 seg a que el usuario elija
        ;same => n,Hangup()     ;Sino se pulsa nada en ese tiempo colgamos -> en vez de esto vamos recurrir a la extensión t

exten => 1,1,NoOp(Opción 1)
        same => n,Playback(ha-seleccionado)
        same => n,SayNumber(1)
        same => n,Goto(labredes,600,1)  ;Redirige la llamada al grupo600

exten => 2,1,NoOp(Opción 2)
        same => n,Playback(ha-seleccionado)
        same => n,SayNumber(2)
        same => n,Goto(labredes,700,1)

exten => i,1,NoOp(Opción inválida) ;extensión i se ocupa de las opciones inválidas
        same => n,Playback(invalid)
        same => n,Goto(s,1)     ;Opción incorrecta, vuelve al principio

exten => t,1,NoOp(timeout)      ;extensión t se ocupa de los timeout
        same => n,Set(a=$[ ${a} + 1]) ;aumentamos la variable a cada vez que se produce un timeout
        same => n,NoOp(a= ${a}) ;verificamos el valor de la variable en la consola
        same => n,Playback(vm-sorry) ;lo siento, no comprendí su respuesta
        same => n,GotoIf($[ ${a} < 2]?s,loop) ;si han pasado 2 timeouts (de 20 seg), redirigimos la llamada por descarte a alguien
        same => n,System(python3 /home/silviu/texto_a_mp3.py 'Paso su llamada con un agente' '02')
        same => n,System(ffmpeg -y -i /tmp/nombre-02.mp3 -ar 8000 -ac 1 /tmp/nombre-02.wav)
        same => n,Playback(/tmp/nombre-02)
        same => n,Goto(labredes,900,1)  ;Llama al grupo de agentes
```

Los audios para el menú han sido grabados con la aplicación **Audacity**, a continuación, explicaré los pasos para utilizarlos en Asterisk:

- Mediante el comando **sox** convertimos el audio .wav a .gsm: **sox ivr-departamentos.wav -r 8000** (ratio -> 8000hz de frecuencia) **-c1** (un canal mono) **ivr-departamentos.gsm**
- Movemos dicho audio a **/usr/share/Asterisk/sounds/es**
- A la hora de llamar el audio, mediante **Background()** o **Playback(),** se debe hacer sin extensión

Llamadas grupales

En **confbridge.conf** realizamos la configuración de las llamadas grupales. Definimos el contexto para el administrador de la llamada grupas y para los usuarios que se unirán, con sus respectivas contraseñas. Activaremos la música de espera cuando la sala esté vacía, avisamos de los nuevos usuarios conectados y el número máximo de miembros en la llamada será de 5:

```
[general]

[admin]
type=user
pin=1234
marked=yes
admin=yes
music_on_hold_when_empty=yes
announce_user_count=yes

[default_user]
type=user
pin=7777
marked=yes
music_on_hold_when_empty=yes
announce_user_count=yes

[default_bridge]
type=bridge
max_members=5
```

En el dialplan utilizaremos la extensión 1000 para conectarnos como administrador y 1001 como usuario invitado. La función **ConfBridge()** se encargará de reconocer a cada uno pasándole como parámetros el nombre del contexto de la llamada grupal (**default_bridge**) y el tipo de usuario que debe comprobar la contraseña:

```
; Invitado a llamada grupal
exten => 1001,1,Progress()
        same => n,Wait(1)
        same => n,ConfBridge(1,default_bridge,default_user)

; Admin de la llamada grupal
exten => 1000,1,Progress()
        same => n,Wait(1)
        same => n,ConfBridge(1,default_bridge,admin)
```

Configuración Avanzada

Integración con MySQL

En este apartado integraremos la centralita con MySQL para que almacene toda la información del registro CDR (Call Detail Record), que almacena todos los detalles de llamadas que se realizan a través de Asterisk.

Por tanto, crearemos una base de datos en MySQL llamada asterisk con una tabla llamada cdr. Para almacenar los datos del cdr correctamente he utilizado un esquema obtenido en internet:

```
CREATE TABLE cdr (
        calldate datetime NOT NULL default '0000-00-00 00:00:00',
        clid varchar(80) NOT NULL default '',
        src varchar(80) NOT NULL default '',
        dst varchar(80) NOT NULL default '',
        dcontext varchar(80) NOT NULL default '',
        channel varchar(80) NOT NULL default '',
        dstchannel varchar(80) NOT NULL default '',
        lastapp varchar(80) NOT NULL default '',
        lastdata varchar(80) NOT NULL default '',
        duration int(11) NOT NULL default '0',
        billsec int(11) NOT NULL default '0',
        disposition varchar(45) NOT NULL default '',
        amaflags int(11) NOT NULL default '0',
        accountcode varchar(20) NOT NULL default '',
        uniqueid varchar(32) NOT NULL default '',
        userfield varchar(255) NOT NULL default ''
);
```

En primer lugar, descargamos el MySQL Connector/ODBC, que se encarga de hacer posible el acceso a los datos desde cualquier aplicación a MySQL. Lo descomprimimos y guardamos el driver libmyodbc82.so.

Configuramos el archivo de configuración de texto para drivers **/etc/odbcinst.ini**:

```
[MySQL]
Description = MySQL Driver
Driver = /usr/lib/x86_64-linux-gnu/odbc/libmyodbc8w.so
```

En el archivo de configuración **/etc/odbc.ini** que almacena las definiciones de todos los DSN de ODBC, especificamos la conexión a la base de datos creada. Llamamos a la conexión con el mismo nombre que la base de datos para que sea sencillo:

```
[asterisk]
Description     = MySQL Connector for Asterisk
Driver          = MySQL
Server          = localhost
Database        = asterisk
User            = root
;Password       = 7777 ;accederemos directamente con (mysql -u root)
```

Ahora que ya hemos establecido la conexión con la base de datos, en **/etc/asterisk/res_odbc.conf** especificamos como vamos a acceder a la tabla dentro de la base de datos. En el contexto **[asterisk]**: **enabled => yes**, **dsn => asterisk**, **pre-connect => yes** y **max_connections => 1**.

Comprobar conexión desde la consola:

```
ubuntuserver*CLI> module reload res_odbc
Module 'res_odbc' reloaded successfully.
    -- Reloading module 'res_odbc.so' (ODBC resource)
[Apr 18 17:57:42] NOTICE[1678]: res_odbc.c:596 load_odbc_config: Registered ODBC class 'asteris
k' dsn->[asterisk]
ubuntuserver*CLI> odbc show asterisk

ODBC DSN Settings
-----------------

  Name:    asterisk
  DSN:     asterisk
    Number of active connections: 1 (out of 1)
```

En **/etc/asterisk/cdr.conf** encontramos la configuración global del registro CDR y en **/etc/asterisk/cdr_adaptive_odbc.conf** indicaremos en que tabla queremos almacenar los datos del CDR y donde empieza:

```
[general]
enabled => yes
dsn => asterisk
max_connections => 1
pre-connect => yes
```

```
[adaptive_connection]
connection = asterisk
table => cdr ;la tabla de la base de datos

alias start => calldate
```

Por último, reiniciamos Asterisk con **service asterisk restart**.

Accediendo a la base de datos en MySQL podemos comprobar que la tabla cdr se carga con todos los datos de las llamadas realizadas:

```
mysql> use asterisk
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-------------------+
| Tables_in_asterisk |
+-------------------+
| cdr               |
+-------------------+
1 row in set (0.00 sec)

mysql> select * from cdr;
+---------------------+------------------------+----------+------+-------------------+-----------------------+-----------------------+------------+----------------------
------+----------+---------+-------------+----------+-------------+-------------------+-----------+
| calldate            | clid                   | src      | dst  | dcontext          | channel               | dstchannel            | lastapp    | lastdata
      | duration | billsec | disposition | amaflags | accountcode | uniqueid          | userfield |
+---------------------+------------------------+----------+------+-------------------+-----------------------+-----------------------+------------+----------------------
------+----------+---------+-------------+----------+-------------+-------------------+-----------+
| 2021-04-05 10:36:28 | "" <pepe>              | pepe     | 601  | labredes          | SIP/pepe-00000000     | SIP/silviu-00000001   | Dial       | SIP/silviu,15,tT
      |        8 |       2 | ANSWERED    |        3 |             | 1617618988.0      |           |
| 2021-04-07 11:42:49 | "" <usuario1>          | usuario1 | 900  | labredes          | SIP/usuario1-00000000 | SIP/agente1-00000001  | Queue      | agentes,,,,50
      |       90 |      90 | ANSWERED    |        3 |             | 1617795769.0      |           |
| 2021-04-07 11:43:39 | "" <usuario2>          | usuario2 | 900  | labredes          | SIP/usuario2-00000002 | SIP/agente1-00000003  | Queue      | agentes,,,,50
      |       49 |      49 | NO ANSWER   |        3 |             | 1617795819.2      |           |
| 2021-04-07 11:44:34 | "" <usuario2>          | usuario2 | 900  | labredes          | SIP/usuario2-00000002 | SIP/agente1-00000004  | Queue      | agentes,,,,50
      |       10 |      10 | ANSWERED    |        3 |             | 1617795819.2      |           |
| 2021-04-07 11:46:23 | "" <agente1>           | agente1  | 202  | labredes          | SIP/agente1-00000005  |                       | Hangup     |
      |        0 |       0 | ANSWERED    |        3 |             | 1617795983.8      |           |
| 2021-04-07 11:48:19 | "" <agente1>           | agente1  | 201  | labredes          | SIP/agente1-00000006  |                       | Hangup     |
      |        0 |       0 | ANSWERED    |        3 |             | 1617796099.10     |           |
| 2021-04-07 11:49:11 | "" <agente1>           | agente1  | 201  | labredes          | SIP/agente1-00000007  |                       | Hangup     |
      |        0 |       0 | ANSWERED    |        3 |             | 1617796151.12     |           |
| 2021-04-07 11:49:59 | "" <agente1>           | agente1  | 202  | labredes          | SIP/agente1-00000008  |                       | Hangup     |
      |        0 |       0 | ANSWERED    |        3 |             | 1617796199.14     |           |
| 2021-04-18 11:42:00 | "" <pepe>              | pepe     | 900  | labredes          | SIP/pepe-00000000     | SIP/agente1-00000001  | Queue      | agentes,,,,120
      |       66 |      66 | NO ANSWER   |        3 |             | 1618746120.0      |           |
| 2021-04-18 11:43:12 | "" <pepe>              | pepe     | 900  | labredes          | SIP/pepe-00000000     | SIP/agente2-00000002  | Queue      | agentes,,,,120
      |       15 |      15 | NO ANSWER   |        3 |             | 1618746120.0      |           |
| 2021-04-18 11:43:32 | "" <pepe>              | pepe     | 900  | labredes          | SIP/pepe-00000000     | SIP/agente1-00000003  | Queue      | agentes,,,,120
```

Integración de agentes

Para este apartado rescato lo aprendido sobre colas y he creado un grupo de agentes. Sin embargo, en el contexto de este grupo no declaro miembros, ya que los miembros se añadirán y eliminarán de forma dinámica. Esto tiene sentido si pensamos en que los agentes son empleados cuya extensión solo interesa que esté disponible para el grupo de agentes cuando estén en su puesto de trabajo.

La estrategia utilizada será **leastrecent**, que llama a la extensión que ha sido menos utilizada por esta cola en un tiempo reciente. De esta forma buscamos comunicar con el agente menos cargado:

```
[agentes]

strategy = leastrecent
;member => SIP/agente1
```

```
; Agentes, teleoperadores
exten => 900,1,NoOp(Agentes)
        same => n,Set(CHANNEL(musicclass)=music_espera)
        same => n,Answer() ;Si no se contesta entra música en espera
        same => n,Queue(agentes,,,,120) ;Sin opciones, URL,etc..solo timeout
        same => n,Hangup
```

Los agentes tendrán a su disposición dos menús, uno para registrarse en el grupo (mediante **AddQueueMember()**) y otro para salir del grupo (mediante **RemoveQueueMember()**). Al entrar al menú deberán elegir el número referente a su puesto para hacer efectiva la acción:

```
exten => 250,1,Goto(ivr-añadir-agentes,s,1) ;Menú para añadir agentes
exten => 251,1,Goto(ivr-remove-agentes,s,1) ;Menú que echa agentes
```

```
[ivr-añadir-agentes]
exten => s,1,Answer
        same => n,System(python3 /home/silviu/texto_a_mp3.py 'Pulsa el número de tu puesto para acceder al grupo de agentes' '04')
        same => n,System(ffmpeg -y -i /tmp/nombre-04.mp3 -ar 8000 -ac 1 /tmp/nombre-04.wav)
        same => n,Playback(/tmp/nombre-04)
        same => n,WaitExten(20)

exten => 1,1,NoOp(Opción 1 => añadir agente1)
        same => n,Answer()
        same => n,AddQueueMember(agentes,SIP/agente1)
        same => n,Playback(beep)
        same => n,Hangup

exten => 2,1,NoOp(Opción 2 => añadir agente2)
        same => n,Answer()
        same => n,AddQueueMember(agentes,SIP/agente2)
        same => n,Playback(beep)
        same => n,Hangup

exten => i,1,NoOp(Opción inválida)
        same => n,Playback(invalid)
        same => n,Goto(s,1)     ;Opción incorrecta, vuelve al principio

exten => t,1,NoOp(timeout)      ;extensión t se ocupa de los timeout
        same => n,Playback(vm-sorry) ;lo siento, no comprendí su respuesta
        same => n,Goto(s,1)
```

```
[ivr-remove-agentes]
exten => s,1,Answer
        same => n,System(python3 /home/silviu/texto_a_mp3.py 'Pulsa el número de tu puesto para desconectarte del grupo de agentes' '05')
        same => n,System(ffmpeg -y -i /tmp/nombre-05.mp3 -ar 8000 -ac 1 /tmp/nombre-05.wav)
        same => n,Playback(/tmp/nombre-05)
        same => n,WaitExten(20)

exten => 1,1,NoOp(Opción 1 => remove agente1)
        same => n,Answer()
        same => n,RemoveQueueMember(agentes,SIP/agente1)
        same => n,Playback(beep)
        same => n,Hangup

exten => 2,1,NoOp(Opción 2 => remove agente2)
        same => n,Answer()
        same => n,RemoveQueueMember(agentes,SIP/agente2)
        same => n,Playback(beep)
        same => n,Hangup

exten => i,1,NoOp(Opción inválida)
        same => n,Playback(invalid)
        same => n,Goto(s,1)     ;Opción incorrecta, vuelve al principio

exten => t,1,NoOp(timeout)      ;extensión t se ocupa de los timeout
        same => n,Playback(vm-sorry) ;lo siento, no comprendí su respuesta
        same => n,Goto(s,1)
```

La funcionalidad de conversión de texto a voz será explicada en el siguiente punto.

Conversión de texto a voz

Para este apartado he utilizado un código en Python que hace uso de la librería **gtts** (Google Text-to-Speech). El programa recoge el texto a convertir y un número con el que identificaremos el archivo que obtenemos como resultado. Especificamos el lenguaje al que

queremos convertir y guardamos el archivo en el directorio **/tmp** con el número pin que hemos indicado y en formato mp3.

```
# Import the required module for text
# to speech conversion
from gtts import *

# This module is imported so that we can
# play the converted audio
import os
import sys
nombre = sys.argv[1]
pin = sys.argv[2]

# The text that you want to convert to audio
mytext = nombre

# Language in which you want to convert
language = 'es'

# Passing the text and language to the engine,
# here we have marked slow=False. Which tells
# the module that the converted audio should
# have a high speed
myobj = gTTS(text=mytext, lang=language, slow=False)
# Saving the converted audio in a mp3 file named
# welcome
myobj.save("/tmp/nombre-" + pin + ".mp3")
```

En el menú mostrado en el apartado anterior podemos ver el funcionamiento de esta funcionalidad:

```
[ivr-añadir-agentes]
exten => s,1,Answer
        same => n,System(python3 /home/silviu/texto_a_mp3.py 'Pulsa el número de tu puesto para
 acceder al grupo de agentes' '04')
        same => n,System(ffmpeg -y -i /tmp/nombre-04.mp3 -ar 8000 -ac 1 /tmp/nombre-04.wav)
        same => n,Background(/tmp/nombre-04)
        same => n,WaitExten(20)
```

Mediante la función **System()** podemos ejecutar comandos como si estuviéramos en la terminal. De esta forma llamamos a la función **texto_a_mp3.py** que contiene el código mostrado previamente, pasándole el texto a convertir a voz y un número pin para reconocer el archivo que obtendremos como resultado.

Utilizaremos también el comando **ffmpeg** para convertirlo a formato **wav**, ya que es el formato que soporta asterisk. Al igual que con el comando **sox**, establecemos la frecuencia de muestreo a 8000hz y un único canal de audio.

Con **Playback()** cargamos dicho archivo de audio pasándoselo como parámetro. Sin embargo, utilizaremos **Background()**, para poder elegir la opción mientras escuchamos el audio.

Recordemos que nuestro programa, y el comando ffmpeg, guardan el archivo en el directorio **/tmp**, por tanto, el parámetro que le pasamos a **Background()** será: **/tmp/nombre-pin** (sin la extensión).

Reconocimiento de voz

Para el reconocimiento de voz, procederemos de una forma similar al apartado anterior. Utilizaremos un programa sencillo en Python que se encargará del reconocimiento de voz. La explicación del código viene comentada en el mismo:

```
import speech_recognition as sr #Librería de reconocimiento de voz de python
import time
import unidecode
import sys

pin = sys.argv[1]

r = sr.Recognizer() #Instancia del speech_recognition

with sr.AudioFile("/tmp/opt-" + pin + ".wav") as source: #Recibe como parámetro el archivo
de voz que va a pasar a texto
    audio = r.listen(source) #Escuchamos el audios

    try:
        text = r.recognize_google(audio, language='es-ES') #Proceso de reconocimiento a tra
vés de google => Google Web Speech API
        time.sleep(1.5) #Le damos el tiempo suficiente para que termine el proceso
        print(unidecode.unidecode(text.replace(' ',''))) #Va a imprimir (tiene como salida)
 el audio convertido en texto => unidecode quita los espacios y acentos para realizar luego
 la comparación
    except:
        print("No entiendo")
```

Ahora, podremos analizar el menú IVR final que he implementado en la práctica:

```
[ivr-1departamento]

exten => s,1,NoOp(IVR 1)        ;No hay opción, simplemente muestra por la consola de comnados Asterisk lo expuesto
        same => n,Answer
        same => n,Set(a=0)
        same => n(loop),Background(ivr-departamentos)
        same => n,WaitExten(20) ;Esperamos 20 seg a que el usuario elija
        ;same => n,Hangup()     ;Sino se pulsa nada en ese tiempo colgamos -> en vez de esto vamos recurrir a la extensión t

exten => 1,1,NoOp(Opción 1)
        same => n,Playback(ha-seleccionado)
        same => n,SayNumber(1)
        same => n,Goto(labredes,600,1)  ;Redirige la llamada al grupo600

exten => 2,1,NoOp(Opción 2)
        same => n,Playback(ha-seleccionado)
        same => n,SayNumber(2)
        same => n,Goto(labredes,700,1)

exten => i,1,NoOp(Opción inválida) ;extensión i se ocupa de las opciones inválidas
        same => n,Playback(invalid)
        same => n,Goto(s,1)     ;Opción incorrecta, vuelve al principio
```

Hasta aquí el menú es igual al mostrado en apartados anteriores. Dependiendo de la opción elegida se contacta con un departamento u otro y la extensión i se encarga de comunicar al usuario que la opción elegida no es válida y le devuelve al principio del menú.

```
exten => t,1,NoOp(timeout)      ;extensión t se ocupa de los timeout
        same => n,Set(a=$[ ${a} + 1]) ;aumentamos la variable a cada vez que se produce un timeout
        same => n,NoOp(a= ${a}) ;verificamos el valor de la variable en la consola
        same => n,Playback(vm-sorry) ;lo siento, no comprendí su respuesta
        same => n,GotoIf($[ ${a} < 2]?s,loop) ;si han pasado 2 timeouts (de 20 seg), redirigimos la llamada por descarte a alguien (p.eje un teleoperador)
        same => n,System(python3 /home/silviu/texto_a_mp3.py 'Para hablar con un agente, diga agente. Para salir del menú, diga salir' '03')
        same => n,System(ffmpeg -y -i /tmp/nombre-03.mp3 -ar 8000 -ac 1 /tmp/nombre-03.wav)
        same => n,Playback(/tmp/nombre-03)

        same => n(inicio),Playback(beep) ;Señal de comienzo de grabación
        same => n,Record(/tmp/opt-1.wav,,4) ;Grabo el audio, doy 4 segundos
        ;same => n,Playback(/tmp/opt-1)
        same => n,Set(result=${SHELL(python3 /home/silviu/voice.py '1')}) ;SHELL es igual que la función system, pero me deja guardar el resultado
        same => n,NoOp(Result = '${result:0:-1}' con longitud ${LEN(${result:0:-1})}) ;Veo el resultado y longitud en consola => borro el primer caracter empezando por
el final "\0?"
        same => n,GotoIf($["${result:0:-1}" = "agente"]?agente) ;Si result = agente => voy a la etiqueta agente
        same => n,GotoIf($["${result:0:-1}" = "salir"]?salir:no) ;Si result = salir => voy a la etiqueta salir, de lo contrario voy a la etiqueta no

        same => n(agente),System(python3 /home/silviu/texto_a_mp3.py 'Paso su llamada con un agente' '02')
        same => n,System(ffmpeg -y -i /tmp/nombre-02.mp3 -ar 8000 -ac 1 /tmp/nombre-02.wav)
        same => n,Playback(/tmp/nombre-02)
        same => n,Goto(labredes,900,1)  ;Llama al grupo de agentes

        same => n(salir),Playback(vm-goodbye)
        same => n,NoOp(Salir del ivr)
        same => n,Hangup()

        same => n(no),Playback(vm-sorry)
        same => n,GoTo(inicio) ;Vuelvo a la etiqueta de grabación
```

Pero ahora, al no tomar una decisión después de los dos intentos concedido al usuario se le indicará (mediante la funcionalidad TTS) que indique mediante reconocimiento de voz si quiere comunicarse con un agente (llama al grupo de agentes) o salir del menú.

Indicaciones adicionales a parte de la explicación en los comentarios del dialplan:

- **SHELL** actúa de una forma parecida a la función **System()**, ya que nos permite ejecutar comandos como si estuviéramos en terminal. Sin embargo, junto a la función **Set()** nos permite almacenar el resultado de lo que hayamos realizado en terminal, en este caso el texto resultado del reconocimiento de voz.
- A **voice.py** debemos pasarle como parámetro el mismo número que el utilizado para identificar el archivo que hemos grabado con **Record()**. En el código ya nos encargamos de reconocer el archivo de audio siempre que tenga la estructura **opt-{IDENTIFICADOR}.wav** y el identificador coincida.
- Al comparar el resultado con las palabras que nos interesan utilizamos **${result:0:-1}**. Lo que estamos haciendo es borrar el primer carácter empezando por el final, ya que el proceso de reconocimiento nos devuelve el texto con un carácter al final que nos impide comparar de forma correcta ("\0"?).

Integración con otra PBX

Dada la situación actual de crisis sanitaria causada por el COVID-19, en vez de conectar con la centralita de otro grupo de laboratorio, clonaremos nuestra máquina virtual y conectaremos con la centralita de la máquina clonada resultante. En VirtualBox, con botón derecho sobre la máquina a clonar, elegimos la opción **"clonar"**, **"Generar nuevas direcciones MAC para todos los adaptadores de red"** como política de dirección MAC y **"Clonación completa"**.

Ahora, al clonar nuestra máquina virtual, contamos con dos centralitas en nuestro ordenador. Sin embargo, utilizarán la misma dirección IP, para cambiar la dirección IP de una de las máquinas utilizaremos el comando **dhclient**. Mediante este comando seguido de la interfaz de red renovamos la dirección IP de la máquina.

Una vez asignada una dirección IP diferente creamos un enlace SIP bidireccional entre ambas centralitas en **sip.conf**, especificando las direcciones de cada una y creamos las extensiones que estarán relacionadas:

```
; Llamadas hacia y desde el PBX_B

[PBX_B]
type=peer
host=192.168.1.114
disallow=all
allow=alaw
context=entrantes_PBX_B

; Extensión 6601
[6601](usuario)
username=6601
secret=7777

; Extensión 6602
[6602](usuario)
username=6602
secret=7777
```

```
; Llamadas hacia y desde PBX_A

[PBX_A]
type=peer
host=192.168.1.133
disallow=all
allow=alaw
context=entrantes_PBX_A

; Extensión 7701
[7701](usuario)
username=7701
secret=7777

; Extensión 7702
[7702](usuario)
username=7702
secret=7777
```

En el dialplan las llamadas de una centralita a otra se sacan por el enlace creado cuando cumplen el patrón de llamada indicado. De PBX_A a B si la extensión empieza por 7 seguida de 3 números, y al revés si empieza por 6 seguido de 3 números. En el caso de las llamadas entrantes definimos el comportamiento para los números correspondientes a cada centralita dentro del contexto **entrantes_PBX_A** o **entrantes_PBX_B**, en cada caso.

```
[labredes]

; Llamadas internas dentro de PBX_A

exten => _6XXX,1,Dial(SIP/${EXTEN})
        same => n,Hangup()

; Salientes hacia PBX_B

exten => _7XXX,1,Answer
        same => n,Wait(1)
        same => n,Dial(SIP/PBX_B/${EXTEN})
        same => n,Hangup()

; Entrantes desde el PBX_B

[entrantes_PBX_B]

exten => 6601,1,Dial(SIP/6601)
        same => n,Hangup()

exten => 6602,1,Dial(SIP/6602)
        same => n,Hangup()
```

```
[labredes]
; Lamadas internas dentro de PBX_B

exten => _7XXX,1,Dial(SIP/${EXTEN})
        same => n,Hangup()

; Salientes hacia PBX_A

exten => _6XXX,1,Answer
        same => n,Wait(1)
        same => n,Dial(SIP/PBX_A/${EXTEN})
        same => n,Hangup()

; Entrantes desde PBX_B

[entrantes_PBX_A]

exten => 7701,1,Dial(SIP/7701)
        same => n,Hangup()

exten => 7702,1,Dial(SIP/7702)
        same => n,Hangup()
```