IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

TP - ToIP SIP & Asterisk

F2B407B

2018-2019

1 Introduction

SIP (*Session Initiation Protocol*) est un protocole de signalisation dédié à l'établissement et la gestion des sessions multimédia. SIP est normalisé dans le RFC 3261 (qui rend obsolète le RFC 2543), et est complété par le RFC 3265 (l'extension SIMPLE pour la gestion de présence et la messagerie instantané). SIP ne transporte pas les données échangées durant la session, pour cela on utilise généralement le protocole RTP (*Real-time Transport Protocol*).

L'architecture typique SIP nécessite bien évidemment des terminaux comme les téléphones Grandstream qui nous utiliserons, ou bien des *softphones* comme Ekiga, Linphone, Kphone, Twinkle, X-Lite, et bien d'autres. Ces terminaux ont la capacité de dialoguer entre eux directement. C'est pour cela que l'on dit parfois que le protocole SIP (comme H323) est un protocole paire-à-paire, contrairement à d'autres comme MGCP ou SSCP. Cependant, il est souvent nécessaire d'introduire dans l'architecture des équipements qui permettrons aux participants de s'atteindre plus facilement, et notemment les fameux *proxy SIP*. Parmis les solutions classiques on peut noter Asterisk (`http://www.asterisk.org/`) ou SER (*SIP Express Router*, et ses descendants : OpenSIPS `http://www.opensips.org/`, Kamailio `http://www.kamailio.org/`, SIP-Router `http://sip-router.org/`). Notons qu'Asterisk peut faire office de passerelle pour signalisation (p.ex : SIP/H323) et la voix (VoIP/PSTN), et messagerie vocale. En revanche, SER apparait comme un proxy-registrar dans la plus pure tradition SIP, c-a-d. il ne gère que la signalisation ; aussi on lui adjoint parfois le gestionnaire de messagerie vocale SEMS `http://www.iptel.org/sems` (qui inclue également un Back-2-Back User Agent (B2BUA) pouvant faire office de Session Border Controller (SBC)).

Nous aborderons[1] ici la solution Asterisk. Asterisk peut être vu comme un auto-commutateur (PABX) logiciel. La vocation première d'Asterisk était justement de gérer des cartes d'extension pour PC offrant des ports FXO (pour se raccorder à une ligne téléphonique) et des ports FXS (pour raccorder des combinés téléphoniques). Il s'agit des cartes Digium, avec leur driver Dahdi (anciennement appelé Zaptel). D'autres cartes sont supportées comme les cartes BRI (Numéris). Le logiciel Asterisk assurant donc l'interconnexion de ces différents média pour transporter les appels. Un PC ayant bien souvent une carte réseau, le support pour la téléphonie par internet était tout naturel. Asterisk s'interconnecte donc avec des équipements VoIP SIP et H323 (et même Skinny), ainsi qu'avec d'autres Asterisk au travers du protocole propriétaire (mais publié) IAX.

Les utilisateurs *s'enregistrent* sur le serveur Asterisk, étape nécessaire à l'établissement de la communication. Asterisk propose également des service comme la messagerie vocale ou la conversation à plusieurs.

1. Remerciements : les manipulations proposées ici sont inspirées d'un tutoriel de Mickael Cissé `mickael.cisse@gmail.com`

2 Mise en place du matériel

Chacune des trois tables de test de la salle de TP est organisée selon la figure 1.

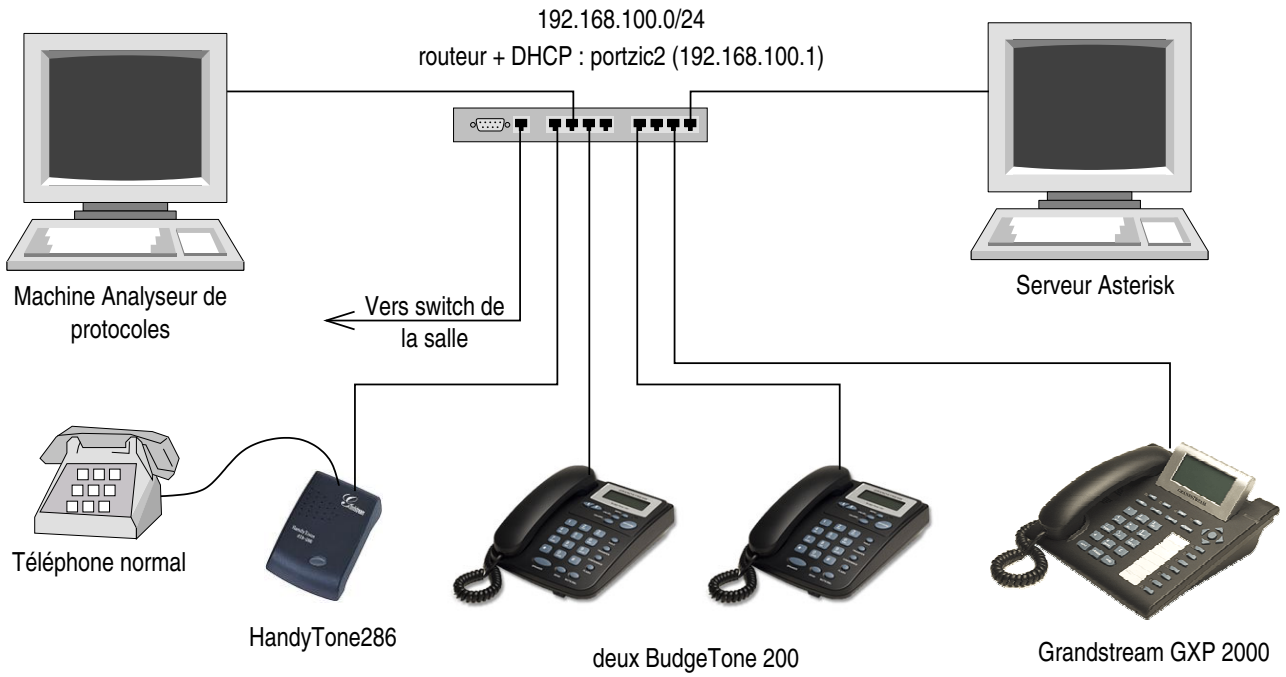

FIGURE 1 – Table de test

Le matériel par banc de test :
— un PC doté du logiciel Asterisk [2]
— un hub
— 3 terminaux VoiIP SIP :
  — Bugde Tone 200
    `http://www.grandstream.com/pdf/BT200UserManual.pdf`
  — Grandstream GXP2000
    `http://www.grandstream.com/pdf/GXP2000UsersManual.pdf`
  — un telephone classique relié à un adptateur ATA Grandstream Handytone ATA286
    `http://www.grandstream.com/pdf/HandyTone-386UserManual.pdf`
— câbles réseaux
— un second PC avec Wireshark pour analyser le réseau

Assurez-vous que tous les équipements sont correctement connectés ensemble sur le hub et alimentez les si ce n'est déjà fait.

3 Configuration du PC Asterisk

— Ouvrez une session sur le PC Asterisk
— Ouvrez un terminal-shell sous le compte `root` (i.e. administrateur). Le mot de passe vous sera communiqué en séance.
— Récupérer l'adresse IP du PC Asterisk (commande `ifconfig`), elle vous servira pour provisionner les téléphones.

2. Nous utiliserons ici le paquet Debian. Une liste de LiveCD Asterisk : `http://www.voip-info.org/wiki-Asterisk+Bootable+CDROM`

4 Configuration manuelle des terminaux VoIP

Trouver la configuration réseaux des terminaux VoIP. Un serveur DHCP (Portzic2) fournit des adresses IP des téléphones IP.

- GXP2000 : l'IP est indiquée à l'écran
- BudgeTone 200 : Trouvez l'adresse IP en faisant `menu` / `IP Addr`
- Téléphone analogique sur boitier HandyTone-286 : décrochez le combiné, appuyez 3 fois sur `*` sur le téléphone ou appuyez sur le bouton rouge du boitier, et composer le `02` sur le téléphone, et finalement écouter l'adresse.

5 Configuration web des terminaux VoIP

Lancez un navigateur web sur l'un des PC. Comme URL, indiquez successivement l'adresse IP de chacun des terminaux.

- Mot de passe `admin`
- Renseigner `SIP serveur` avec l'IP de la machine Asterisk
- Utiliser les numéros téléphoniques 800, 801, 802, 803 pour `SIP User ID` (pour le BT200, le GXP2000, et le ATA286)
- Utiliser les mêmes numéros pour `Authenticate ID` (le login)
- Choisir un mot de passe (par exemple `"1234"`). Attention à ne pas confondre le mot de passe d'utilisateur SIP et le mot de passe d'administration du terminal.
- Fixer le *mode DTMF* sur `rfc2833`, pour une meilleur interropérabilité.

5.1 Connexion au CLI d'Asterisk

Le daemon Asterisk n'a pas été lancé automatiquement au boot du PC. Pour le lancer, dans un terminal `root` sur le PC, tapez la commande `/etc/init.d/asterisk start`

Pour interagir avec ce daemon, nous avons un genre de *shell* dédié appelé CLI (*Command Line Interface*).

- Ouvrez une fenêtre terminal-shell
- Si besoin, passez root par `sudo su -`
- Lancer le CLI par `asterisk -vvvcr`

Quelques commandes intéressantes :

- `reload`
- `sip show peers`
- `core show channels`
- `core show channel SIP/800-`*`xyz`*
- `core show applications`
- `core show application answer`
- `quit`

```
linux2:~#  asterisk -vvvvcr
Asterisk 1.4.21.2~dfsg-2, Copyright (C) 1999 - 2008 Digium, Inc. and others.
Created by Mark Spencer <markster@digium.com>
== Parsing '/etc/asterisk/asterisk.conf': Found
== Parsing '/etc/asterisk/extconfig.conf': Found
Connected to Asterisk 1.4.21.2~dfsg-2 currently running on linux2 (pid = 30888)
Verbosity is at least 4
linux2*CLI>
```

6 Déclaration des terminaux VolP SIP

Asterisk a plusieurs *canaux* (ou *trunk*), un pour chacun des protocoles réseau supportés (SIP, H323, IAX, etc.), plus ceux pour des cartes Digium ou autre (Dahdi, etc.). Nous allons nous intéresser au canal SIP.

Consultez le fichier `/etc/asterisk/sip.conf`. Il a déjà été préparé avec notamment les contextes qui nous intéressent, celles qui déclarent chacun des clients SIP (nos téléphones Grandstream). Nous retrouvons donc :

```
[800]
type=friend             ; Télépone "amis"
secret=password         ; Définition d'un mot de passe
username=800            ; Numéro du poste
host=dynamic            ; Adresse IP fournie par DHCP (et donc non connue ici)
canrenvite=yes          ; Permet les transferts en utilisant un re-invite
dtmfmode=auto           ; Le mode DTMF à utiliser
context=from-sip        ; L'endroit du dialplan où les appels vont arriver
```

Définir de même les autres téléphones. Quand cela est fait, demandez à Asterisk de recharger les différents fichiers de configuration avec la commande `reload` de la console CLI.

Il faut ensuite vérifier que les terminaux VoIP s'enregistrent correctement. (Il peut s'avérer nécessaire de redémarrer les terminaux pour les forcer à se réenregistrer.) Cela peut se faire en regardant les logs d'Asterisk dans le CLI. On peut également le vérifier à tout moment avec la commande `sip show peers` :

```
linux2*CLI> sip show peers
Name/username              Host            Dyn Nat ACL Port     Status
803/803                    192.168.100.91   D          5060     Unmonitored
802/802                    192.168.100.60   D          5060     Unmonitored
801/801                    192.168.100.71   D          5060     Unmonitored
800/800                    192.168.100.81   D          5060     Unmonitored
4 sip peers [Monitored: 0 online, 0 offline Unmonitored: 4 online, 0 offline]
```

6.1 Captures réseau

À l'aide de `wireshark` sur le second PC, observez les échanges de messages SIP. Repérez notamment la phase d'authentification. (Provoquez-la au besoin : soit en faisant `Reload` dans le panneau de configuration, soit débranchez - rebranchez l'alimentation d'un téléphone.)

7 Création du Dialplan

Le plan de numérotation, et finalement tout le comportement d'Asterisk lorsque l'on fait un appel, est décrit dans le fichier `/etc/asterisk/extensions.conf`.

Vous savez, lorsque vous appelez le numéro d'une entreprise (ou de tout établissement dont la téléphonie est géré par un PABX), vous tapez quelques chiffres au clavier de votre téléphone pour naviguer dans les menus d'une boite vocale, ou simplement pour composer le numéro de poste de votre correspondant. Ces chiffres que vous tapez après le numéro de téléphone de l'entreprise constituent l'*extention* (jargon américain). D'où le nom de ce fichier si important, car Asterisk sait faire tout ça.

Ouvrez ce fichier `/etc/asterisk/extensions.conf`. Il y a un contexte spécial `[from-sip]` avec un dialplan minimum pour basculer sur le trunk SIP lorsqu'il y a un appel sur des numéros en $8XX$.

```
...

[macro-call]
; ${ARG1} is the number to call
...
exten => s,2,Dial(SIP/${ARG1},10)

...

[from-sip]
exten => _8XX,1,NoOp
exten => _8XX,2,NoOp
exten => _8XX,3,Macro(call,${EXTEN})
exten => i,1,Hangup ; invalid
exten => t,1,Hangup ; timeout
```

Ceci est la syntaxe exacte (ne pas remplacer `XX` par des nombres !). Cette section permettra de gérer tous les appels arrivant dans le contexte "`[from-sip]`" qui est celui que nous avons indiqué lors de la déclaration des terminaux.

Cette section fait appel à une macro définie un peu plus haut, et au final tout appel d'un numéro de la forme `8XX` engendrera un appel *SIP/8XX*.

Note : si vous modifiez la configuration, utilisez la commande `reload` du CLI pour la recharger.

8 Quelques scénarios de test

Maintenant que la configuration de base est en place, nous allons l'essayer.

- Tester avec les téléphones : appel simple, la mise en attente, la conférence à 3, et le transfert (avec ou sans consultation)
- Utiliser `core show channels` pour visualiser l'état des appels
- Systématiquement, avec `wireshark` suivez le déroulement des appels : les messages `INVITE` avec leur SDP, les réponses `OK`, les addresses IP (du correspondant, du serveur ?) et numéros de ports. Repérez par où passent les paquets RTP.

8.1 Pas de communication

```
linux2*CLI> core show channels
Channel              Location             State   Application(Data)
0 active channels
0 active calls
```

8.2 Pemier appel

Le 800 appelle le 801, laisse sonner une fois, puis raccroche (touche "`#`" ou "`Send`" pour valider le numéro)

```
linux2*CLI>
-- Executing [801@from-sip:1] NoOp("SIP/800-087d4010", "") in new stack
-- Executing [801@from-sip:2] NoOp("SIP/800-087d4010", "") in new stack
-- Executing [801@from-sip:3] Macro("SIP/800-087d4010", "call,801") in new stack
-- Executing [s@macro-call:1] GotoIf("SIP/800-087d4010", "0?50") in new stack
-- Executing [s@macro-call:2] Dial("SIP/800-087d4010", "SIP/801,10") in new stack
-- Called 801
-- SIP/801-087e1e30 is ringing
== Spawn extension (macro-call, s, 2) exited non-zero on 'SIP/800-087d4010' in macro 'c
== Spawn extension (macro-call, s, 2) exited non-zero on 'SIP/800-087d4010'
```

En utilisant Wireshark (sur un second PC éventuellement), prenez une trace réseau d'un appel SIP. Trouvez dans la trace réseau le numéro appelé et le numéro appelant (dans le message `INVITE`).

8.3 Second appel

Le 800 appelle le 801, qui prend l'appel quelques instants et raccroche

```
linux2*CLI>
-- Executing [801@from-sip:1] NoOp("SIP/800-087d4010", "") in new stack
-- Executing [801@from-sip:2] NoOp("SIP/800-087d4010", "") in new stack
-- Executing [801@from-sip:3] Macro("SIP/800-087d4010", "call,801") in new stack
-- Executing [s@macro-call:1] GotoIf("SIP/800-087d4010", "0?50") in new stack
-- Executing [s@macro-call:2] Dial("SIP/800-087d4010", "SIP/801,10") in new stack
-- Called 801
-- SIP/801-087e1e30 is ringing
-- SIP/801-087e1e30 answered SIP/800-087d4010
-- Native bridging SIP/800-087d4010 and SIP/801-087e1e30
```

```
linux2*CLI> core show channels
Channel              Location             State   Application(Data)
SIP/801-087e1e30     (None)               Up      Bridged Call(SIP/800-087d4010)
SIP/800-087d4010     s@macro-call:2       Up      Dial(SIP/801,10)
2 active channels
1 active call
```

En utilisant Wireshark, quel est le codec utilisé ?

Combien de paquets RTPs sont envoyés par seconde ? (Utiliser la fonction "*IoGraph*" de Wireshark.) Afficher le call flow avec Wireshark (*Telephony/VoIP calls/Graph*).

Comment voit-on que c'est 801 qui raccroche et non 800 ?

8.4 Mise en attente et conférence à 3

Le 801 appelle le 800, le met en attente, appelle le 802, puis conférence à 3.

Testez ces différents services, de préférence sur le poste GXP2000, c'est le plus simple pour cela. Au besoin (ce n'est pas forcément très intuitif), consultez le *Manuel Utilisateur* du téléphone.

Vous pouvez utiliser le mode main-libre pour que tout le groupe puisse entendre ce qu'il se passe.

801 appelle le 800

```
linux2*CLI>
-- Executing [800@from-sip:1] NoOp("SIP/801-087d4010", "") in new stack
-- Executing [800@from-sip:2] NoOp("SIP/801-087d4010", "") in new stack
-- Executing [800@from-sip:3] Macro("SIP/801-087d4010", "call,800") in new stack
-- Executing [s@macro-call:1] GotoIf("SIP/801-087d4010", "0?50") in new stack
-- Executing [s@macro-call:2] Dial("SIP/801-087d4010", "SIP/800,10") in new stack
-- Called 800
-- SIP/800-087e1e30 is ringing
-- SIP/800-087e1e30 answered SIP/801-087d4010
-- Native bridging SIP/801-087d4010 and SIP/800-087e1e30
```

```
linux2*CLI> core show channels
Channel              Location             State   Application(Data)
SIP/800-087e1e30     (None)               Up      Bridged Call(SIP/801-087d4010)
SIP/801-087d4010     s@macro-call:2       Up      Dial(SIP/800,10)
2 active channels
1 active call
```

801 met 800 en attente (*HOLD*)

```
linux2*CLI>
    -- Started music on hold, class 'default', on SIP/800-087e1e30
```

Avec Wireshark, essayez de repérer par quels messages SIP/SDP le téléphone 801 signale qu'il met l'appel en attente.

801 appel 802 (sur la *LINE2*)

```
linux2*CLI>
-- Executing [802@from-sip:1] NoOp("SIP/801-087ff260", "") in new stack
-- Executing [802@from-sip:2] NoOp("SIP/801-087ff260", "") in new stack
-- Executing [802@from-sip:3] Macro("SIP/801-087ff260", "call,802") in new stack
-- Executing [s@macro-call:1] GotoIf("SIP/801-087ff260", "0?50") in new stack
-- Executing [s@macro-call:2] Dial("SIP/801-087ff260", "SIP/802,10") in new stack
-- Called 802
-- SIP/802-088031d8 is ringing
-- SIP/802-088031d8 answered SIP/801-087ff260
-- Native bridging SIP/801-087ff260 and SIP/802-088031d8
```

```
linux2*CLI> core show channels
Channel              Location             State   Application(Data)
SIP/802-088031d8     (None)               Up      Bridged Call(SIP/801-087ff260)
SIP/801-087ff260     s@macro-call:2       Up      Dial(SIP/802,10)
SIP/800-087e1e30     (None)               Up      Bridged Call(SIP/801-087d4010)
SIP/801-087d4010     s@macro-call:2       Up      Dial(SIP/800,10)
4 active channels
2 active calls
```

801 fait une conférence à 3 (*CONF*) avec 800 (*LINE1*)

```
linux2*CLI>
-- Stopped music on hold on SIP/800-087e1e30
```

```
linux2*CLI> core show channels
Channel              Location             State   Application(Data)
SIP/802-088031d8     (None)               Up      Bridged Call(SIP/801-087ff260)
SIP/801-087ff260     s@macro-call:2       Up      Dial(SIP/802,10)
SIP/800-087e1e30     (None)               Up      Bridged Call(SIP/801-087d4010)
SIP/801-087d4010     s@macro-call:2       Up      Dial(SIP/800,10)
4 active channels
2 active calls
```

8.5 Transfert d'appel

801 appel 800

```
linux2*CLI>
-- Executing [800@from-sip:1] NoOp("SIP/801-087d4010", "") in new stack
-- Executing [800@from-sip:2] NoOp("SIP/801-087d4010", "") in new stack
-- Executing [800@from-sip:3] Macro("SIP/801-087d4010", "call,800") in new stack
-- Executing [s@macro-call:1] GotoIf("SIP/801-087d4010", "0?50") in new stack
-- Executing [s@macro-call:2] Dial("SIP/801-087d4010", "SIP/800,10") in new stack
-- Called 800
-- SIP/800-087e1e30 is ringing
-- SIP/800-087e1e30 answered SIP/801-087d4010
-- Native bridging SIP/801-087d4010 and SIP/800-087e1e30
```

```
linux2*CLI> core show channels
Channel              Location             State   Application(Data)
SIP/800-087e1e30     (None)               Up      Bridged Call(SIP/801-087d4010)
SIP/801-087d4010     s@macro-call:2       Up      Dial(SIP/800,10)
2 active channels
1 active call
```

800 transfert à 802 (*HOLD*, puis *TRANSFER*)

```
linux2*CLI>
-- Started music on hold, class 'default', on SIP/801-087d4010
-- Stopped music on hold on SIP/801-087d4010
== Spawn extension (from-sip, 802, 0) exited non-zero on 'SIP/801-087d4010' in macro 'c
== Spawn extension (from-sip, 802, 0) exited non-zero on 'SIP/801-087d4010'
-- Executing [802@from-sip:1] NoOp("SIP/801-087d4010", "") in new stack
-- Executing [802@from-sip:2] NoOp("SIP/801-087d4010", "") in new stack
-- Executing [802@from-sip:3] Macro("SIP/801-087d4010", "call,802") in new stack
-- Executing [s@macro-call:1] GotoIf("SIP/801-087d4010", "0?50") in new stack
-- Executing [s@macro-call:2] Dial("SIP/801-087d4010", "SIP/802,10") in new stack
-- Called 802
-- SIP/802-087e5da8 is ringing
-- SIP/802-087e5da8 answered SIP/801-087d4010
-- Native bridging SIP/801-087d4010 and SIP/802-087e5da8
```

```
linux2*CLI> core show channels
Channel              Location             State   Application(Data)
SIP/802-087e5da8     (None)               Up      Bridged Call(SIP/801-087d4010)
SIP/801-087d4010     s@macro-call:2       Up      Dial(SIP/802,10)
2 active channels
1 active call
```

Dans Wireshark, repérez la requête SIP `REFER` et regardez son contenu. Puis, repérez les deux requêtes `INVITE` initiées par Asterisk. Suivez également les `NOTIFY`.

9 Fonctionnalités Avancées

Maintenant que la prise en main est faite, nous allons étudier quelques fonctions plus sophistiquées.

9.1 Enregistrement d'un mémo vocal

Nous ajoutons la possibilité d'enregistrer un fichier audio (fonction *record*) en appelant le `10X`. (Terminez votre message par la touche `#`.)

Nous avons également la possibilité de jouer ce fichier (fonction *playback*) en appelant le `11X`.

Notez que dans ce scénario, on ne dispose concrètement que de dix *mémo* possibles, que n'importe qui peut déposer et/ou consulter. Ce n'est pas un fonctionnement de répondeur téléphonique habituel.

— Dans `extensions.conf`

```
[from-sip]
.................

; Enregistrer un son
exten => _10X,1,record(/tmp/rec_${EXTEN:2}.wav,,,k)
exten => _10X,2,Hangup

; Rejouer un son
exten => _11X,1,playback(/tmp/rec_${EXTEN:2})
exten => _11X,2,Hangup
```

Utiliser la commande `core show application Record` pour avoir des informations.

9.2 Renvoi sur une messagerie vocale

Nous ajoutons un renvoi sur messagerie vocale en cas d'occupation du destinataire, et consultation de sa boîte vocale en appelant le 888 (mot de passe 1234).

— On définit une messagerie vocale pour chaque poste dans `/etc/asterisk/voicemail.conf`

Dans `voicemail.conf`

```
[default]
800 => 1234,800,root@localhost
801 => 1234,801,root@localhost
802 => 1234,802,root@localhost
803 => 1234,803,root@localhost
```

— Dans le dialplan (`extensions.conf`), on définir le renvoi à la messagerie vocale (commande `Voicemail`) en cas d'occupation.

Ainsi, quand un poste ne répond pas au bout de 10 secondes ou qu'il est déjà occupé, l'appel doit-être enregistré sur la messagerie vocale.

Dans `extensions.conf`

```
[macro-call]
; ${ARG1} is the number to call
....
exten => s,2,Dial(SIP/${ARG1},10)
exten => s,3,NoOp(status=${DIALSTATUS})
exten => s,4,GotoIf($["${DIALSTATUS}" = "BUSY"]?100)
exten => s,5,GotoIf($["${DIALSTATUS}" = "CHANUNAVAIL"]?110)
exten => s,6,GotoIf($["${DIALSTATUS}" = "NOANSWER"]?110)
exten => s,7,GotoIf($["${DIALSTATUS}" = "CONGESTION"]?110)
exten => s,8,Hangup

...

exten => s,100,Voicemail(${ARG1}@default,b)
exten => s,101,Hangup

exten => s,110,Voicemail(${ARG1}@default,u)
exten => s,111,Hangup
```

— Bien sûr, l'utilisateur absent doit pouvoir consulter sa propre messagerie en appelant un numéro (le 888 par exemple) (commande `VoiceMailMain`).

Dans `extensions.conf`

```
[from-sip]
...
; Consulter la messagerie vocale
exten => 888,1,VoiceMailMain(${CALLERID(num)})
exten => 888,1,Hangup
```

9.3 Renvoi d'appel

Il s'agit ici de programmer une redirection d'appel lorsqu'un collaborateur doit s'absenter. Dans le cas présent, le collaborateur ayant le poste 801 compose le 13802. Dès lors, tout appel vers le 801 sera automatiquement redirigé vers le poste 802. Bien sûr, cela doit fonctionner quelque soit le numéro de téléphone, pas seulement avec 802. (On utilise pour cela la petite base de données intégrée à Asterisk.)

À son retour, le collaborateur appelle le 13 pour effacer son renvoi.

On implémentez ce scénario à l'aide des fonctions `DB`, `DB_EXISTS`, `DB_DELETE`.

Il faut également penser à gérer le cas où le poste 801 compose le 13801, et que quelqu'un appelle ensuite ce poste... Un limiteur de boucle est vivement conseillé !

Dans `extensions.conf`

```
[macro-setforward]
; ${ARG1} is the number to redirect to
exten => s,1,Set(DB(CF/${CALLERID(NUM)})=${ARG1})

[macro-call]
; ${ARG1} is the number to call
exten => s,1,GotoIf(${DB_EXISTS(CF/${ARG1})}?50)
...
exten => s,50,Dial(SIP/${DB(CF/${ARG1})},10)
exten => s,51,Hangup

...

; Point d'entree pour SIP
[from-sip]
...
; Pose un renvoi
exten => _13XXX,1,Macro(setforward,${EXTEN:2})
exten => _13XXX,2,Playback(vm-goodbye)
exten => _13XXX,3,Hangup

; Efface le renvoi d'appel
exten => _13,1,Set(old_val=${DB_DELETE(CF/${CALLERID(NUM)})})
exten => _13,2,Playback(vm-goodbye)
exten => _13,3,Hangup
```

En cours d'utilisation tapez dans asterisk : `database show`

En cas de bug , si jamais vous avez l'impression que votre base de donnée se comporte de manière douteuse suite à vos divers essais, il se peut que la base soit corrompue. Dans ce cas (dans ce cas seulement), vous pouvez supprimer le fichier `/var/lib/asterisk/astdb` puis relancez Asterisk `/etc/init.d/asterisk restart`

10 Interconnexion d'Asterisk

Maintenant que l'on a vu quelques services de base sur Asterisk, il serait intéressant de passer des appels d'un Asterisk à un autre. On peut utiliser les protocole IAX ou SIP.

Dans le cadre de ce TP, nous allons utiliser SIP.

10.1 Création des trunks

Dans le fichier de configuration SIP, on crée autant de `trunk` que de serveur Asterisk à joindre (c'est un compte SIP comme pour un téléphone classique).

Dans `sip.conf` :

```
[trunk1]
type=friend
secret=1234
username=trunk1
fromuser=trunk1
host=192.168.100.12
canreinvite=yes
dtmfmode=rfc2833
context=from-sip

[trunk2]
type=friend
secret=1234
username=trunk2
fromuser=trunk2
host=192.168.100.15
canreinvite=yes
dtmfmode=rfc2833
context=from-sip

[trunk3]
type=friend
secret=1234
username=trunk3
fromuser=trunk3
host=192.168.100.18
canreinvite=yes
dtmfmode=rfc2833
context=from-sip
```

10.2 Routage des trunks

On complète le dialplan avec des numéros permettant d'appeler les différents Asterisk. Par exemple le 01800 pour appeler le poste 800 de l'Asterisk 1, etc.

Dans `extensions.conf` :

```
; Acceder a un autre Asterisk
exten => _01.,1,NoOp
exten => _01.,2,Dial(SIP/${EXTEN:2}@trunk1)
exten => _01.,3,Hangup

exten => _02.,1,NoOp
exten => _02.,2,Dial(SIP/${EXTEN:2}@trunk2)
exten => _02.,3,Hangup

exten => _03.,1,NoOp
exten => _03.,2,Dial(SIP/${EXTEN:2}@trunk3)
exten => _03.,3,Hangup
```