TELEPHONIE SUR IP

Cissé Alioune

Lemaire Yann

Regnier David

Razafindrabe Livantsoa 4 RT
2008

PLAN

Introduction

I. VoIP : Généralités

II. Choix d’Asterisk

III. Installation d’Asterisk et des softphones

IV. Configuration minimale d’Asterisk et des softphones

V. Configuration avancée

Conclusion

Introduction

La Voix sur IP ou Voice over IP (VoIP) est une technique en pleine émergence qui permet de communiquer par la voix via l'Internet ou tout autre réseau acceptant le protocole TCP/IP. Cette technologie est notamment utilisée pour supporter le service de Téléphonie IP (ToIP pour Telephony over Internet Protocol). Elle constitue un tournant dans le monde de la communication. En effet, la convergence du triple Play (voix, données et vidéo) fait partie aujourd'hui des enjeux principaux des acteurs de la télécommunication.

Comme toute innovation technologique qui se respecte, la VoIP doit non seulement simplifier le travail mais aussi faire économiser de l'argent. Les entreprises dépensent énormément en communications téléphoniques, or le prix des communications de la VoIP est dérisoire en comparaison. En particulier, plus les interlocuteurs sont éloignés, plus la différence de prix est intéressante. De plus, la téléphonie sur IP utilise jusqu'à dix fois moins de bande passante que la téléphonie traditionnelle. Ceci apporte donc de grand intérêt pour la voix sur réseau privé.

Vu l'importance de la VoIP, nous avons décidé de l'étudier et de la mettre en pratique dans le cadre du bureau d'étude Multimédia.

I. Présentation de la Voix sur IP

Pour être plus précis, le signal numérique obtenu par numérisation de la voix est découpé en paquets qui sont transmis sur un réseau IP vers une application qui se chargera de la transformation inverse (des paquets vers la voix). Au lieu de disposer à la fois d'un réseau informatique et d'un réseau téléphonique commuté (RTC), l'entreprise peux donc, grâce à la VoIP, tout fusionner sur un même réseau. En effet, la téléphonie devient de la "data". Les nouvelles capacités des réseaux à haut débit devraient permettre de transférer de manière fiable des données en temps réel. Ainsi, les applications de vidéo ou audioconférence ou de téléphonie vont envahir le monde IP qui, jusqu'alors, ne pouvait raisonnablement pas supporter ce genre d'applications (temps de réponse important, gigue, Qos,...). Plus que la latence, c'est la gigue qui pénalise la voix sur IP. En effet, s'il y a des fluctuations du signal en amplitude et fréquence, il faudra un mécanisme de remise en ordre des paquets afin de restituer le message vocal, processus qui se traduira par des blancs et des attentes.

Il apparaît donc clairement que dans le sillage de cette avancée technologique, les opérateurs, entreprises ou organisations et fournisseurs devaient introduire de nouveaux services traitant la voix pour bénéficier de l'avantage du transport unique IP. C'est ainsi que les premières technologies de VoIP imaginées étaient propriétaires et donc très différentes les unes des autres. Pourtant, un système qui est censé mettre des gens et des systèmes en relation exige une certaine dose de standardisation. C'est pourquoi sont apparus des protocoles standards comme le H323 issu de l'organisation de standardisation européenne ITU-T sur la base de la signalisation voix RNIS et SIP (Session Initiation Protocol) standardisé et normalisé par l'IETF (Internet Engineering Task Force). Ces deux protocoles sont utilisés pour l'interactivité en temps réel.

Dans le cadre de ce bureau d'étude, nous avons utilisé SIP. Ce choix s'explique par le fait que SIP s'inspire du protocole Http alors que H.323 s'inspire de la téléphonie et est plus modulaire et peut fonctionner avec d'autres protocoles. Par conséquent, il est donc plus souple que H.323. L'autre raison est liée au fait que la documentation concernant H.323 est difficile d'accès car l'ITU fait payer les droits d'accès aux derniers développements de cette technologie, en dehors des efforts faits par le projet OpenH.323 pour rendre cette technologie accessible à tous. Cet ensemble de normes du H.323 ne s'avèrent pas toujours compatibles avec d'autres protocoles à cause de son développement inspiré de la téléphonie, ce qui peut rendre son utilisation un peu "rigide".

Nous allons donc par la suite vous présenter une description du protocole SIP.

a. Protocole SIP

SIP est un protocole de signalisation appartenant à la couche application du modèle OSI. Son rôle est d'ouvrir, modifier et libérer les sessions.

L'ouverture de ces sessions permet de réaliser de l'audio ou vidéoconférence, de l'enseignement à distance, de la voix (téléphonie) et de la diffusion multimédia sur IP essentiellement. Un utilisateur peut se connecter avec les utilisateurs d'une session déjà ouverte. Pour ouvrir une session, un utilisateur émet une invitation transportant un descripteur de session SDP (Session Description Protocol) permettant aux utilisateurs souhaitant communiquer de s'accorder sur la compatibilité de leur média (voix, vidéo, données...) et des paramètres de communication. SIP permet donc de relier des stations mobiles en transmettant ou redirigeant les requêtes vers la position courante de la station appelée.

SIP se charge aussi de

- l'authentification,
- la localisation des multiples participants, de l'analyse de leurs profils et ressources,
- vérifier la disponibilité des correspondants c'est-à-dire si le terminal appelé souhaite communiquer et autorise l'appelant à le contacter,
- l'établissement et suivi d'appel : avertir les parties appelante et appelée de la demande d'ouverture de session, gérer le transfert et la fermeture des appels,
- gérer les fonctions évoluées : cryptage, retour d'erreur

Enfin, SIP possède l'avantage de ne pas être attaché à un médium particulier et est sensé être indépendant du protocole de transport des couches basses. Ce qui fait que tout type de données et de protocoles peut être utilisé pour cet échange.

Avec SIP, les utilisateurs qui ouvrent une session peuvent communiquer en mode :

- point à point : communication entre deux machines, on parle d'unicast
- diffusif : plusieurs utilisateurs en multicast, via une unité de contrôle M.C.U (Multipoint Control Unit)
- combinatoire (mixte des modes 2 précédents): plusieurs utilisateurs pleinement interconnectés en multicast via un réseau à maillage complet de connexions.

Puis que le but que nous nous sommes fixés est d'établir une communication téléphonique entre deux clients, nous allons travailler en mode point à point dans ce BE.

Les échanges entre un terminal appelant et un terminal appelé se font par l'intermédiaire de requêtes :

- INVITE indique que l'application ou utilisateur correspondante à l'Url SIP spécifié est invité à participer à une session. Le corps du message décrit cette session (par ex : médias supportés par l'appelant). En cas de réponse favorable, l'invité doit spécifier les médias qu'il supporte.
- ACK permet de confirmer que le terminal appelant a bien reçu une réponse définitive à une requête Invite.

- OPTIONS : un proxy server en mesure de contacter l'UAS (terminal) appelé doit répondre à une requête Options en précisant ses capacités à contacter le même terminal.
- BYE est utilisée par le terminal de l'appelé afin de signaler qu'il souhaite mettre un terme à la session.
- CANCEL est envoyée par un terminal ou un proxy server afin d'annuler une requête non validée par une réponse finale comme, par exemple, si une machine ayant été invitée à participer à une session, et ayant accepté l'invitation ne reçoit pas de requête Ack, alors elle émet une requête Cancel.
- REGISTER est utilisée par le client pour enregistrer l'adresse listée dans l'Url par le serveur auquel il est relié

Une réponse à une requête est caractérisée par un code appelé code d'état et un motif.

Un code d'état est un entier codé sur 3 bits indiquant un résultat à l'issue de la réception d'une requête. Ce résultat est précisé par une phrase, textbased (UTF-8) expliquant le motif du refus ou de l'acceptation de la requête. Le code d'état est donc destiné à l'automate gérant l'établissement des sessions SIP et les motifs aux programmeurs. Il existe 6 classes de réponses et donc de codes d'état, représentées par le premier bit :

- 1xx = Information, la requête a été reçue et continue à être traitée.
- 2xx = Succès, l'action a été reçue avec succès, comprise et acceptée.
- 3xx = Redirection, une autre action doit être menée afin de valider la requête.
- 4xx = Erreur du client, la requête contient une syntaxe erronée ou ne peut pas être traitée par ce serveur.
- 5xx = Erreur du serveur, le serveur n'a pas réussi à traiter une requête apparemment correcte.
- 6xx = Echec général, la requête ne peut être traitée par aucun serveur.

La mise en place d'un système SIP nécessite deux types de composantes, les users agents (UAS, UAC) et un réseau de serveurs :

- UAS (User Agent Server) représentant l'agent de la partie appelée. C'est une application de type serveur qui contacte l'utilisateur lorsqu'une requête SIP est reçue et elle renvoie une réponse au nom de l'utilisateur.
- L'UAC (User Agent Client) représentant l'agent de la partie appelante. C'est une application de type client qui initie les requêtes.

Le relais mandataire ou PS (Proxy Server), auquel est relié un terminal fixe ou mobile, agit à la fois comme un client et comme un serveur. Un tel serveur peut interpréter et modifier les messages qu'il reçoit avant de les retransmettre :

- Le RS (Redirect Server) réalise simplement une association (mapping) d'adresses vers une ou plusieurs nouvelles adresses par exemples lorsqu'un client appelle un terminal mobile, il y a redirection vers le PS le plus proche et en mode multicast, le message émis est redirigé vers toutes les sorties auxquelles sont reliés les destinataires). Notons qu'un Redirect Server est consulté par l'UAC comme un simple serveur et ne peut émettre de requêtes contrairement au PS.

- Le LS (Location Server) fournit la position courante des utilisateurs dont la communication traverse les RS et PS auxquels il est rattaché. Cette fonction est assurée par le service de localisation.
- Le RG (Registrar) est un serveur qui accepte les requêtes Register et offre également un service de localisation comme le LS. Chaque PS ou RS est généralement relié à un Registrar.

Notons que l'utilisation de SIP présente des avantages :

- Protocole rapide grâce à la séparation des champs d'en-tête et du corps du message facilitant le traitement des messages et diminuant ainsi leur temps de transition dans le réseau.
- Nombre des en-têtes est limité (36 au maximum et en pratique, moins d'une dizaine d'en-têtes sont utilisées simultanément) allégeant du coup l'écriture et la lecture des requêtes et réponses.
- De plus, il sépare les flux de données de ceux de la signalisation, ce qui rend plus souple l'évolution "en direct" d'une communication (arrivée d'un nouveau participant, changement de paramètres...).

Dans ce BE, nous avons utilisé un ordinateur portable qui va jouer le rôle du Proxy Serveur. Ce qui a nécessité l'installation du logiciel libre Asterisk qui transforme un ordinateur en un commutateur téléphonique privé ou PABX.

b. Le logiciel Astérisk

La télécommunication, bien qu'elle subisse un essor incroyable actuellement, fait partie des technologies qui ne sont pas encore touchés par le phénomène du « libre ». En effet, la plupart des produits industriels de la télécommunication restent non accessibles au grand public. Ceci est dû à leur prix qui est excessivement cher d'une part et à leur complexité, leur incompatibilité avec d'autres systèmes et le fait qu'ils soient vite obsolètes d'autre part.

C'est sur ce point qu'Asterisk met en exergue son plus grand avantage : il offre une grande liberté de développement de fonctionnalité et de comportement qui lui donne une grande compatibilité avec les différentes technologies.

Cette grande flexibilité a un prix qui n'est autre que la difficulté de configuration, non pas parce qu'il est confus, illogique ou crypté mais au contraire parce qu'il est très sensible et modulaire.

Asterisk, c'est quoi…

Un serveur Multiplateforme

Asterisk est un serveur de commutateur privé qui a été développé pour fonctionner sur un large éventail de système d'exploitation, à citer : Linux, Mac OSX, OpenBSD, FreeBsd, Sun Solaris et même Windows.

Un commutateur privé (PBX)

Asterisk intègre des algorithmes de routage et de commutation d'appels téléphonique. Asterisk peut faire office de passerelle du réseau IP vers le réseau commuté publique. En d'autre terme, il permettra aux utilisateurs de la téléphonie IP d'établir une connexion sortante ou entrante vers le réseau téléphonique commuté.

Un serveur multi-protocole

Asterisk supporte un large panel de protocole pour maitriser la transmission de la voix à travers un réseau IP incluant le H.323, la Session Initiation Protocole (SIP), le Media Gateway control Protocol (MGCP) ou le Skinny Client Control Protocol (SCCP).

Un serveur multifonction

Son caractère « libre » a permit aux développeurs et intégrateurs d'étendre largement les fonctionnalités du serveur. Les fonctionnalités sont très diverses et évolutives. Un des exemples illustrant la performance d'Asterisk est de pouvoir fonctionner en tant que serveur central d'appel téléphonique. En se connectant sur internet, il peut offrir une multitude de services aux utilisateurs comme les conditions météorologiques, les transactions bancaires et même demander une exécution de tâche à un ordinateur personnel via un téléphone.

Asterisk comprend un nombre très élevé de fonctions permettant l'intégration complète pour répondre à la majorité des besoins en téléphonie (messagerie vocale, appel, détection de la parole…). Il permet de remplacer totalement, par le biais de cartes FXO/FXS, un PABX propriétaire, et d'y adjoindre des fonctionnalités de VoIP pour le transformer en PBX IP. Il permet également de fonctionner totalement en VoIP, par le biais de téléphones SIP ou IAX du marché, ce qui notre cas. Pour ce faire, nous avons installé le logiciel x-lite qui va simuler le client de téléphonie sur IP depuis un ordinateur. Ce client est appelé softphone.

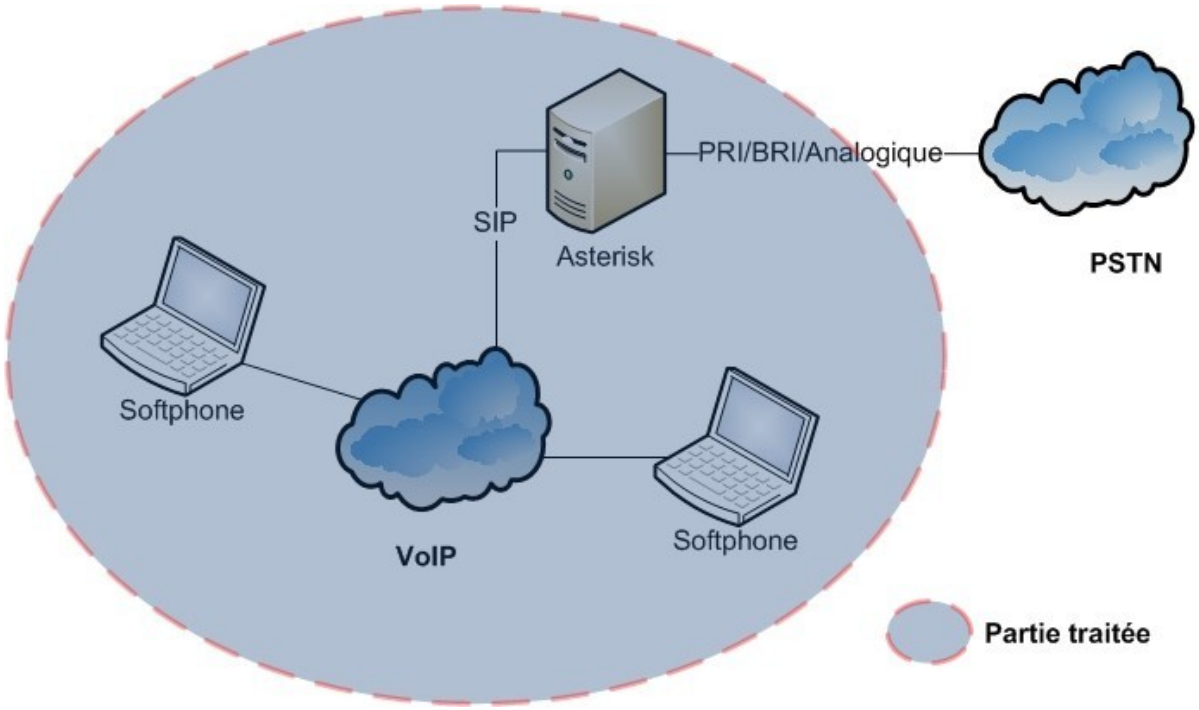
Enfin, des fonctionnalités de routage d'appel, menu vocal et boites vocales, entres autres, le placent au niveau des PBX les plus complexes.

La raison pour laquelle nous avons choisi Asterisk comme serveur de VOIP est de pouvoir bénéficier de toutes les fonctionnalités déjà implémentées. D'une part, compte tenu de la contrainte de temps à laquelle nous étions soumis, cela nous a permis de ne pas nous concentrer sur l'implémentation de la transmission de donnée. D'autre part, cela nous a permis d'acquérir des connaissances dans la configuration d'Asterisk qui se présente comme un serveur doté d'un avenir prometteur.

II. Installation d'Astérisk et des softphones

Notre projet consiste donc à faire l'étude du maximum de fonctionnalités offertes par un PBX Asterisk. Il s'agit ici de traiter uniquement les possibilités d'Asterisk dans le cadre d'une solution VoIP dans un réseau local. Donc toute la partie concernant la connexion de l'IPBX avec le réseau téléphonique classique n'est pas abordée.

Nous avons à notre disposition trois ordinateurs portables, un jouera le rôle du PBX et les deux autres les clients à travers un softphone tel que X-Lite. Le protocole de VoIP utilisé sera SIP (car il a l'avantage d'être celui qui est le plus utilisé sur les équipements VoIP et ses spécifications sont libres d'accès (SIP RFC 2543)). Voici la topologie réseau de notre projet qui y correspond :



Il s'agit ici d'une petite solution de téléphonie VoIP : l'analyse du système requis pour le serveur n'est donc pas indispensable. Cependant voici un tableau des recommandations pour la configuration du serveur en fonction du nombre de canaux (dans notre cas 2 canaux):

Type de système	**Nombre de canaux**	**Fréquence CPU et RAM Min.**
« passe-temps »	< 5	400 MHz x86, 256 MB RAM
SOHO	< 10	1 GHZ x86, 512 MB RAM
Petite activité	< 25	3 GHZ x86, 1 GB RAM
Moyen / important	>25	Plusieurs CPU ou serveurs (architecture distribuée)

Lors de l’analyse du système requis (dans le cas d’une situation exclusivement VoIP), d’autres paramètres principaux sont à prendre à compte tels que le nombre d’appels simultanés potentiels, le « dialplan » (**plan d’appel**) qui est utilisé et le pourcentage de trafic qui requiert l’utilisation de codecs tels que G.729 et GSM. Ici nous n’en tenons pas compte, il ne s’agit pas de dimensionner une solution VoIP à caractère professionnel. Pour information, la configuration de notre serveur est 1.83GHZ, 1 GB RAM.

A. Installation d'Astérisk

Asterisk (1.4) est compatible avec n'importe quelle distribution Linux (en tout cas dont la version du kernel est au moins 2.4.5). Voici la liste des paquets nécessaires à l'installation et au fonctionnement d'un système Asterisk :

Package name	Installation command	Note	Used by
GCC 3.x	`yum install -y gcc`	Required to compile zaptel, libpri, and asterisk	libpri, zaptel, asterisk
ncurses-devel	`yum install -y ncurses-devel`	Required by menuselect	menuselect
libtermcap-devel	`yum install -y libtermcap-devel`	Required by asterisk	asterisk
Kernel Development Headers	`yum install -y kernel-devel`	Required to compile zaptel	zaptel
Kernel Development Headers (SMP)	`yum install -y kernel-smp-devel`	Required to compile zaptel	zaptel
GCC C++ 3.x	`yum install -y gcc-c++`	Required by asterisk	asterisk
OpenSSL (optional)	`yum install -y openssl-devel`	Dependency of OSP, IAX2 encryption, res_crypto (RSA key support)	asterisk
newt-devel (optional)	`yum install -y newt-devel`	Dependency of zttool	zaptel
zlib-devel (optional)	`yum install -y zlib-devel`	Dependency of DUNDi	asterisk
unixODBC; unixODBC-devel (optional)	`yum install -y unixODBC-devel`	Dependency of func_odbc, cdr_odbc, res_config_odbc, res_odbc, ODBC_STORAGE	asterisk
libtool (optional; recommended)	`yum install -y libtool`	Dependency of ODBC-related modules	asterisk
GNU make (version 3.80 or higher) [a]	`yum install -y make`	Required to compile zaptel and asterisk	asterisk

[a] It is a common problem among new installs on some Linux distriebutons to see *GNU make* versions of 3.79 or lower. Note that Asterisk will no longer build correctly unless you have at least version 3.80 of *GNU make*.

Dans notre cas, la distribution CentOS que nous avions installé ne demandait aucune mise à jour.

Nous devons maintenant installer le système avec tous les paquets nécessaires. La plupart des systèmes Asterisk sont composées de trois packages principaux: le programme principal Asterisk, les drivers de téléponie zapata (zaptel) et la librairie PRI (libpri). Il est recommandé d'installer ces trois paquets bien que dans le cadre d'une utilisation purement basée VoIP, seul le premier paquet est réellement essentiel.

Zaptel est nécessaire pour tout hardware analogique ou digital, incluant le driver « ztdummy » permettant de jouer le rôle d'horloge de référence (par exemple pour fournir une synchronisation pour les applications de type « musique en attente »).

Libpri permet l'utilisation d'interfaces PRI ISDN permettant ainsi de communiquer avec des réseaux téléphoniques analogiques.

Un paquet additionnel « asterisk-sounds » est aussi recommandé. Depuis la version 1.4.0 d'Asterisk (décembre 2006), plusieurs formats audio sont supportés tels que G.729 et GSM. Le serveur choisi le format qui requiert le moins de CPU pour transcoder le fichier son.

Nous allons installer ces trois packages par soucis de simplicité (il est toujours possible de désactiver les modules inutiles après l'installation).

Voici donc la démarche que nous avons suivie pour l'installation **d'Asterisk 1.4** et des paquets associés:

1. Télécharger les sources

```
# cd /usr/src/
# wget http://downloads.digium.com/pub/asterisk/asterisk-1.4-current.tar.gz
# wget http://downloads.digium.com/pub/libpri/libpri-1.4-current.tar.gz
# wget http://downloads.digium.com/pub/zaptel/zaptel-1.4-current.tar.gz
```

2. Extraire le code source

Les commandes qui suivent permettent d'extraire les sources dans leurs répertoires respectifs :

```
# cd /usr/src/
# tar zxvf zaptel-1.4-current.tar.gz
# tar zxvf libpri-1.4-current.tar.gz
# tar zxvf asterisk-1.4-current.tar.gz
```

3. Compilation des paquets

Lors de l'installation d'Asterisk, nous avons la possibilité de choisir les modules à installer. Comme certains modules nécessitent d'avoir installé les paquets « zaptel » et « libpri » avant, nous d'abord installer ceux-ci avant Asterisk :

3.1 Compiler zaptel

```
# cd /usr/src/zaptel-version
# make clean
# ./configure
# make menuselect
# make
# make install
# make config
```

« make config » installe un script qui permet de lancer zaptel automatiquement au démarrage de l'ordinateur.

3.2 Compiler libpril

```
# cd /usr/src/libpri-version
# make clean
# make
# make install
```

3.3 compiler asterisk

 # cd /usr/src/asterisk-version
 # make clean
 # ./configure

Dans la version d'Asterisk 1.4, un programme permet de sélectionner quels modules sont à installer avant de le compiler. Cela permet d'éviter d'éditer le makefile manuellement :

 # make menuselect

En fait notre projet consistant à s'approprier le plus de fonctionnalités offertes par ce système, nous allons installer tous les modules possibles.

Ci-dessous l'aperçu du menu de « l'autoconf » :

```
**************************************
      Asterisk Module Selection
**************************************

         Press 'h' for help.

   ---> 1.  Applications
        2.  Call Detail Recording
        3.  Channel Drivers
        4.  Codec Translators
        5.  Format Interpreters
        6.  Dialplan Functions
        7.  PBX Modules
        8.  Resource Modules
        9.  Voicemail Build Options
        10. Compiler Flags
        11. Module Embedding
        12. Core Sound Packages
        13. Music On Hold File Packages
        14. Extras Sound Packages
```

Et finalement:

 # make install
 # make samples
 # make config

« make samples » permet d'installer les fichiers de configuration par défaut.

L'installation d'Asterisk est maintenant terminée.

Nous vérifions maintenant que le serveur se lance avec la commande /usr/sbin/asterisk (lancé en tant que **daemon**). L'on a la possibilité de lancer le serveur avec une interface (terminal) permettant de lancer des commandes (appelé asterisk CLI) : /usr/sbin/asterisk –h (utile pour debugger)

B. Installation des softphones

L'installation des softphones ne requiert aucune manipulation particulière. Nous avons choisi d'installer X-Lite sur les deux machines clientes (sous Windows Vista).

III. Configuration minimale d'Astérisk et des softphones

A. Objectifs et généralités

Notre objectif est maintenant de mettre en place une configuration minimale permettant une communication entre les deux softphones.

Les numéros de téléphones choisis arbitrairement sont 101 et 102 et le protocole VoIP utilisé est SIP.
L'adresse IP du serveur sera 192.168.0.254 et celles des softphones 192.168.0.1 et 2.

Seul le canal SIP est donc utilisé dans ce cas (les canaux FXO, FXS et IAX ne sont donc pas mis en place ici).

Bien que l'objectif de cette première partie soit de vérifier que le système fonctionne correctement, nous allons tout de même regarder les répertoires contenant les fichiers de configuration qui nous intéressent :

- Tous les fichiers de configuration sont dans le répertoire /etc/asterisk excepté « zaptel.conf » pour le paquet du même nom qui est dans /etc/

- Tous les modules d'Asterisk (applications, codecs, formats et canaux) sont dans /usr/lib/asterisk/modules/. Etant donné que nous avons installé tous les modules par défaut, ceux que nous n'utiliserons pas pour notre système final devront être « désactivés » dans le fichier modules.conf (pour ne pas charger inutilement le système).

- Les scripts personnels sont dans le répertoire /var/lib/asterisk/agi-bin. AGI signifie « Asterisk Gateway Interface » et est une interface permettant de faire communiquer des programmes externes avec la plan d'appel (« dialplan ») du système Asterisk.

- Les fichiers sons se trouvent dans les répertoires /var/lib/asterisk/mohmp3/ et /var/lib/asterisk/sounds/.

- Les fichiers de configuration relatifs à la messagerie vocale sont dans /var/spool/asterisk (entre autres).

- Le programme Asterisk et le fichier de configuration associé est dans /var/run (pour la distribution CentOs).

- Les erreurs sont dans le repertoires /var/log/asterisk (toujours important de le signaler (-:).

Afin de comprendre le protocole VoIP et son fonctionnement avec Asterisk, voici un peu de vocabulaire utile :

On appelle extrémité (comprendre “endpoint”) des agents qui sont de deux types: client et serveur. Le client est celui qui génère une requête et le serveur celui qui répond à la requête en générant une réponse. Quand deux agents clients s’appellent (ex : un softphone appelle un autre softphone), on génère des requêtes qui sont envoyées au proxy SIP. Celui-ci détermine à qui cette requête est destinée et l’envoie. Les deux agents négocient ainsi la mise en place d’une communication VoIP. Les données sont ensuite transmises directement via un protocole RTP (« real time protocol »). Les serveurs proxy SIP ne gèrent par le transport des données mais seulement les paquets SIP. **Asterisk, par contre, agit à la fois en tant que client et serveur** (appelé Back-To-Back User Agent: B2BUA). Par exemple, quand un softphone appelle un numéro : la communication est établie entre Asterisk et ce softphone. Asterisk détermine que ce numéro correspond à un autre softphone et établie une seconde communication avec ce second softphone. Du point de vue des softphones, la communication est seulement faite avec Asterisk.

B. Configuration d’Astérisk

Nous devons dans un premier temps enregistrer les 2 softphones SIP dans Asterisk. Dans le fichier /etc/asterisk/sip.conf il faut donc les ajouter :

```
[101]     ;nom/numéro du softphone, il faudra donc configurer le softphone en conséquence

type=friend       ;permet d’entrer à la fois dans le « dialplan » et de le quitter pour
                  ;établir une communication

context=central ;voir plus tard
host=192.168.0.1 ;dynamic est aussi possible
```

Il faut réécrire ces lignes en changeant l’adresse IP et le numéro par 102.

On paramètre maintenant le « dialplan » pour que les téléphones puissent communiquer.

Dans le fichier /etc/asterisk/extensions.conf il faut ajouter:

```
[internal] ; nom de l’ « extension »

exten => 101,1,Verbose(1|Extension 101)
```

```
exten => 101,n,Dial(SIP/101,30)
exten => 101,n,Hangup()


exten => 102,1,Verbose(1|Extension 102)
exten => 102,n,Dial(SIP/102,30)
exten => 102,n,Hangup()

 [central]

Include => internal
```

On ne change rien aux autres extensions par défaut. Nous avions faits « make samples » lors de l'installation d'Asterisk, ce qui facilite la configuration. En effet il faudrait sinon ajouter d'autres extensions comme [incoming] qui correspond à ce que fait Asterisk.

C'est évidement une configuration minimaliste qui est considérée comme non sécurisée et non flexible mais notre objectif est seulement d'établir une communication entre les 2 softphones.

C. Configuration des softphones

Pour configure les 2 softphones (dans notre cas X-Lite), il faut cliquer sur le bouton « settings », sélectionner « system settings » et « sip proxy ». Ce qui permet d'afficher la configuration qui est relativement simple, il suffit de respecter les deux points suivants, laissant le reste par défaut :

- Username : 101 ou 102
- Domain/Realm et SIP proxy : 192.168.0.254

D. Tests effectués

Maintenant il nous reste à tester si l'on peut établir une communication. Comme nous avons changé les fichiers de configuration d'Asterisk, il faut soit relancer le serveur: « **Asterisk restart** » soit relancer les fichiers de configuration à partir du terminal d'Asterisk :

*CLI>
*CLI> **dialplan reload**
*CLI> **sip reload**

Il est possible de vérifier si les deux softphones se sont enregistrés auprès du serveur avec la commande (dans le terminal asterisk):

*CLI> **sip show peers**

Le resultat est alors :

Name/username Host Dyn Nat ACL Port Status
101/101 192.168.1.1 D N 5061 OK (63 ms)
102/102 192.168.1.2 D N 5061 OK (58 ms)
2 sip peers [2 online , 0 offline]

On peut avoir plus de detail sur cet enregistrement avec la commande « sip show peer 101 » par exemple:

* Name : 101
Secret : <Not set>
MD5Secret : <Not set>
Context : central
Subscr.Cont. : <Not set>
Language :
AMA flags : Unknown
Transfer mode: open
CallingPres : Presentation Allowed, Not Screened
Callgroup :
Pickupgroup :
Mailbox :
VM Extension : asterisk
LastMsgsSent : 32767/65535
Call limit : 0
Dynamic : NO
Callerid : "" <>
MaxCallBR : 384 kbps
Expire : 1032
Insecure : no
Nat : RFC3581
ACL : No
T38 pt UDPTL : No
CanReinvite : Yes
PromiscRedir : No
User=Phone : No
Video Support: No
Trust RPID : No
Send RPID : No
Subscriptions: Yes
Overlap dial : Yes
DTMFmode : rfc2833
LastMsg : 0

ToHost :
Addr->IP : 192.168.1.1 Port 5061
Defaddr->IP : 0.0.0.0 Port 5060
Def. Username: 101
SIP Options : (none)

Codecs : 0x8000e (gsm|ulaw|alaw|h263)
Codec Order : (none)
Auto-Framing: No
Status : Unmonitored
Useragent : X-Lite release 1105d
Reg. Contact : sip:101@192.168.1.1:5061

L’on peut maintenant appeler d’un softphone à un autre: la communication fonctionne.

Nous avons effectivement permis l’établissement d’une communication VoIP au travers du PBX Asterisk mais sans comprendre les principes du « dialplan » (avec la signification des contextes, extensions…). L’étape suivante consiste donc à bien comprendre le fonctionnement du « dialplan » et ses fonctionnalités associées et d’essayer de créer un système VoIP plus intéressant.

IV. Configuration Avancées

A. Principes du « dialplan »

Le « dialplan » est en fait le centre de n'importe quel système Asterisk. Il permet de définir comment se comportent les appels entrants ou sortants. Il consiste globalement en une liste d'instructions ou d'étapes qu'Asterisk va suivre. A la différence des systèmes de téléphonie classiques, ce dialplan est entièrement paramétrable.

A. La syntaxe

Comme nous avons pu le voir précédemment, le dialplan se trouve dans le fichier extensions.conf. Il est constitué de quatre principaux concepts : les contextes, les extensions, les priorités et les applications.

Lors de l'installation d'Asterisk, nous avions générer des fichiers de configurations faisant office d'exemples (avec la commande « make samples »). Notre fichier extensions.conf comporte donc déjà des exemples très utiles. Les explications qui suivent sont donc à mettre en relation avec ces ceux-ci :

a. Les contextes

Les dialplans sont découpés en sections appelées contextes: les contextes sont des groupes d'extensions qui ont différents objectifs. Voici les principales caractéristiques des contextes à retenir :

- Ils peuvent interagir entre eux.

- Une extension qui est définie dans un contexte est isolée des autres extensions provenant de contextes différents (à moins qu'une interaction entre eux soit explicitement spécifiée)

- Les contextes sont définis tels quels : [nom_du_contexte] (ex : [central]).

- Le nom du contexte peut être composé de lettres minuscules ou majuscules, de chiffres ou des caractères « _ » et « - ».

- Toutes les instructions qui suivent la définition du contexte font partie de celui-ci.

Par exemple, au début de extensions.conf, le contexte [general] est défini (ne pas le supprimer tous comme le suivant ([globals]), il contient les paramètres généraux du dialplan qui sont généralement laissées tels quels).

Lorsque l'on définit un canal (là où l'on connecte des équipements au système : nos deux softphones dans sip.conf), l'un des paramètres que nous avons définis est le contexte associé ([central]). En fait le contexte est point dans le dialplan où les connexions du canal commencent. Les contextes permettent aussi de fournir une certaine sécurité : l'on peut selon le softphone qui appelle, donner des accès différents (par exemple interdire les appels longues distances dans certaines situations : non traités ici puisque tout est local).

a. Les extensions, priorités et applications

Normalement, le mot extension fait référence à un unique identifiant donnée à une ligne qui permet d'appeler un équipement téléphonique particulier (numéro associé qui correspond au numéro de téléphone de l'équipement qui permet d'établir une communication). Ici une extension peut définir à la fois des extensions téléphoniques (un où un groupe de téléphone) et une unique série d' « étapes » (contenant chacune une application associée). Dans chaque contexte, l'on peut définir autant d'extensions que l'on veut. L'extension, au sens large, définit ce qui arrive selon les appels.

La syntaxe associée est « exten=>nom_extension, priorité, application()».

Dans le cas où l'extension définit le « numéro » d'un équipement téléphonique (où groupe dans le cas d'une conférence), le nom de l'extension peut correspondre, selon la comptabilité avec les protocoles VoIP utilisés, à un numéro, un nom d'utilisateur…

Dans notre cas, bien que SIP permette ses variantes, ce sera un numéro qui sera défini.

Il existe cependant une extension particulière « s » utilisé lorsqu'un appel est émis sans aucune extension précisée : « s » signifie « start » (cela permet d'établir un plan d'appel par défaut).

Voici un exemple permettant de faire commencer tous les appels entrants de la même manière :

```
[incoming]
exten => s,1,application()
exten => s,2,application()
exten => s,3,application()
```

Chaque extension incluant plusieurs étapes, la « priorité » correspond au **numéro d'étape**. Chaque priorité est numérotée séquentiellement en commençant par 1 et exécute l' « application » choisie.

L' « application » (ou commande à exécuter) est l'action qui est effectuée lors de l'appel. Chaque application fait une spécifique action sur la ligne courante comme :

- lire un fichier son

- accepter une entrée DTMF (lorsque l'utilisateur appuie sur une touche de son softphone lors d'une communication)

- Appeler une autre ligne

- Raccrocher la ligne…

Certaines applications sont susceptibles de recevoir des arguments, modifiant ainsi leurs comportements, un équivalent est celui d'une fonction C.

Voici un exemple classique:

```
exten => 102,1,Answer()
```

Le nom de l'extension est 102, sa priorité est 1 et l'application est answer() (répond au téléphone).

La priorité peut aussi correspondre à « n » pour « next », l'exemple qui suit illustre cette situation :

```
exten => 102,1,Answer()
exten => 102,n,do something
exten => 102,n,do something else
exten => 102,n,do one last thing
exten => 102,n,Hangup()
```

Cela permet de faciliter le changement de dialplan (être obligé de numéroter chaque étape peut être source d'erreur).

1. Exemples de dialplans

Voici, pour illustrer les concepts précédents quelques exemples de plans d'appel:

```
[incoming]
exten => s,1,Answer()
exten => s,n,Playback(fichier_son)
exten => s,n,Hangup()
```

Ici on répond à l'appel (associé au contexte « incoming »), ensuite un fichier son est lancé et on termine la communication.

Le chemin du fichier son n'est pas obligatoirement spécifié : il s'agit dans ce cas d'un chemin relatif qui correspond en fait à /var/lib/asterisk/sound/fichier_son**. Il ne faut pas préciser l'extension du fichier son** (exemple : fichier_son**.gsm**) même si plusieurs fichiers son différents portent le même nom. Car comme on l'a vu dans la partie I/, asterisk choisi le fichier qui coûte le moins de temps CPU (et cela dépend de la situation). Quoi qu'il en soit, si l'on précise l'extension du fichier, ça ne fonctionne pas.

Il y a un moyen assez simple pour créer ses propres fichier son au format adapté, voici le dialplan :

```
exten => 101,1,Wait(2)
exten => 101,n,Record(/tmp/fichier_son_101:wav)
exten => 101,n,Wait(1)
exten => 101,n,Playback(/tmp/fichier_son_101)
exten => 101,n,Wait(2)
exten => 101,n,Hangup()
```

Ce dialplan permet de téléphoner du softphone dont le numéro est 101 et d'enregistrer un message vocal. Il est enregistré au format .wav (le moins coûteux pour Asterisk en temps CPU) dans le répertoire /tmp/. Il suffit d'appuyer sur le softphone sur la touche # pour terminer l'enregistrement. Ce message vocal est ensuite répété.

Finalement pour l'utiliser en tant que message d'accueil, il suffit de le déplacer dans le répertoire par défaut (pas obligatoire puisque dans le dialplan on peut spécifier le chemin absolu) avec la commande :

```
mv /tmp/fichier_son_101.wav /var/lib/asterisk/sounds/custom/message_accueil.wav
```

```
[incoming]
exten => 123,1,Answer()
exten => 123,n,Background(main-menu)
exten => 123,n,WaitExten(5)
exten => 1,1,Playback(digits/1)
exten => 2,1,Playback(digits/2)
exten => 3,1,Playback(digits/3)
exten => 4,1,Playback(digits/4)
```

Ici on répond à l'appel dont l'extension est 123, l'application « Background » est ensuite exécutée. Celle-ci permet de lire un fichier son tout en laissant la main à l'utilisateur pour appuyer sur des touches (application DTMF). Par contre lorsque l'utilisateur a effectivement appuyé sur une touche, le fichier son est interrompu. Une autre application « WaitExten » est lancée : elle permet de réceptionner les touches tapées par l'utilisateur. Et finalement si l'utilisateur appuie sur 1,2,3 ou 4, Asterisk va lire ce chiffre.

```
[incoming]
exten => 123,1,Answer()
exten => 123,n,Background(enter-ext-of-person)
exten => 123,n,WaitExten(5)
exten => 1,1,Playback(digits/1)
exten => 1,n,Goto(incoming,123,1)
exten => 2,1,Playback(digits/2)
exten => 2,n,Goto(incoming,123,1)
exten => 3,1,Playback(digits/3)
exten => 3,n,Goto(incoming,123,1)
exten => i,1,Playback(pbx-invalid)
exten => i,n,Goto(incoming,123,1)
exten => t,1,Playback(vm-goodbye)
exten => t,n,Hangup()
```

Dans ce plan d'appel, l'application « Goto » est ajoutée : elle permet de revenir à une extension précisée dans l'appel de celle-ci. Les extensions « i » correspondent à des extensions entrées par l'utilisateur qui ne correspondent à aucune extension (extension dîtes dans ce cas invalide). Les extension « t » correspondent au « timeout » donc lorsque le délai d'attente est dépassé (timeout de WaitExten par défaut est de 5 secondes dans ce plan d'appel).

```
[incoming]
exten => 101,1,Dial(SIP/102,10,m)
exten => 101,n,Playback(vm-nobodyavail)
```

```
exten => 101,n,Hangup()
```

Ici l'application Dial est utilisée : elle permet tout simplement d'appeler. La syntaxe est plus ou complexe, il faut préciser le type de canal, l'identifiant de l'équipement téléphonique que l'on appelle (généralement un numéro mais si le protocole utilisé le permet, pourquoi pas un « username »), le nombre de secondes avant le « timeout » et des options très nombreuses mais facultatives en troisième argument (« m » permet d'écouter de la musique (music on hold : comprendre musique en attente) au lieu des « ring-ring » habituels. D'ailleurs c'est ici que le paquet « zaptel » nous est utile : cette fonctionnalité nécessite une synchronisation en temps (voir partie I/).

En ce qui concerne la configuration de la fonctionnalité « music on hold », le fichier /etc/asterisk/musiconhold.conf précise la location des fichiers son:

```
[default]

mode=files
directory=/var/lib/asterisk/mohwav
random=yes
```

Il faut donc y mettre des .wav à l'aide du dialplan permettant de créer ses propres fichiers (il y a des manipulations particulières à faire si l'on encode des .wav à partir de .mp3, il faudrait utiliser , par exemple, « SoX »… pas traité ici).

Il y a évidement pleins d'autres exemples mettant ainsi en place l'utilisation de variables, de conditions, d'includes, de programmes extérieurs à travers des scripts perl ou php...

B. Objectifs fixés

Nous avons donc vu brièvement dans la partie précédente les concepts du plan d'appel. Il s'agit maintenant de mettre réellement en application des fonctionnalités avancées. Voici donc les principales caractéristiques du plan d'appel que nous avons envisagé :

- Il s'agira de l'équivalent d'un central téléphonique dont on accédera en appelant le 10.

- Différentes options seront possibles en appuyant sur la touche correspondante.

- Les fonctionnalités offertes par ce central seront alors :

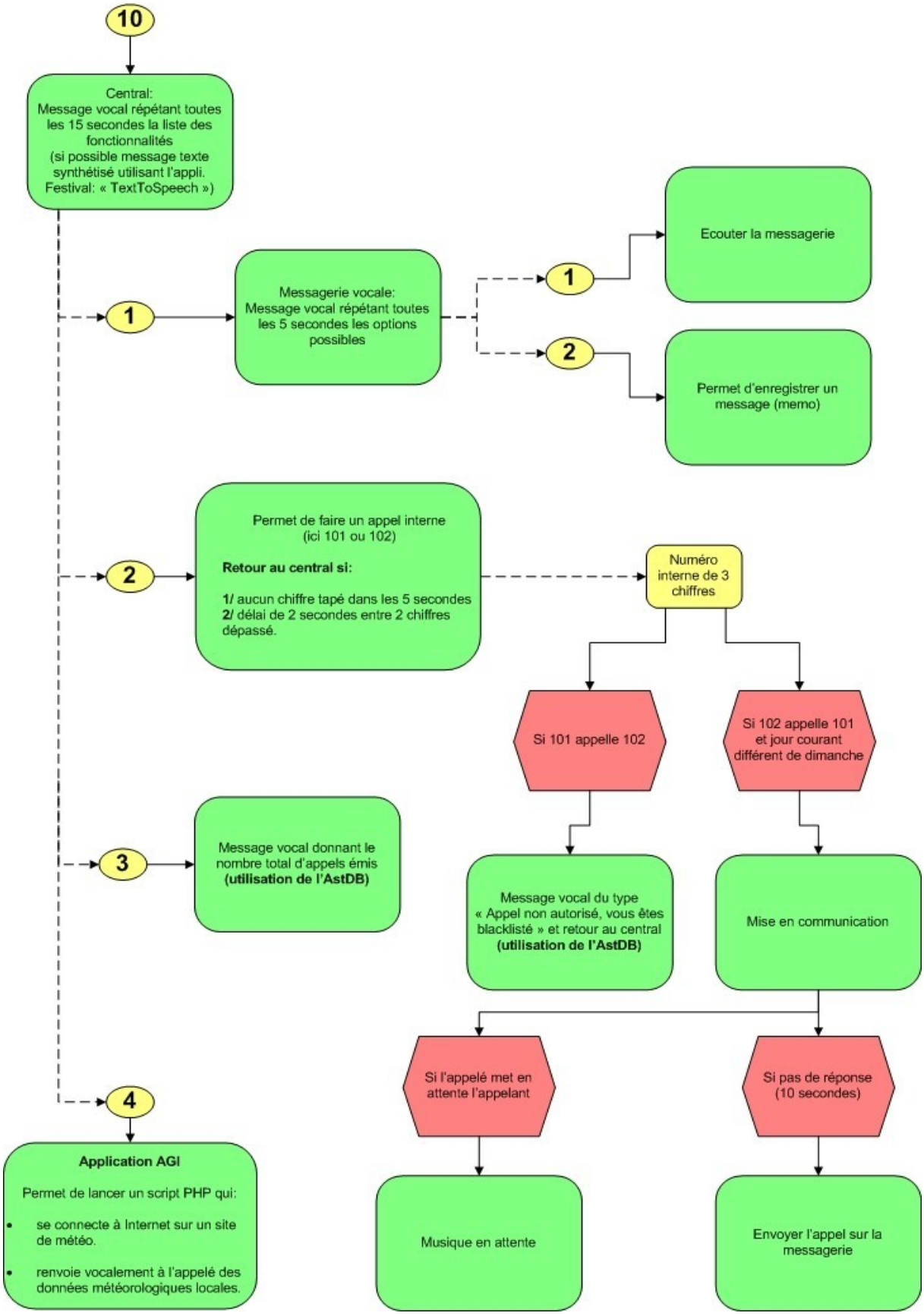
    - Une application de messagerie vocale.

- Un appel interne refusé ou non selon des critères différents avec l’utilisation d’une base de données Mysql.

- Des options permettant de consulter des statistiques d’appel.

- La consultation de données météorologiques provenant d’internet.

Ci-dessous l’illustration de plan d’appel :



C. Configuration du « dialplan

1. Mise en place de la structure du central

```
[central]
Exten => 10,1,Answer()
Exten => 10,n,Background(accueil_central)
Exten => 10,n,WaitExten(15)

Exten => 1,1,Goto(messagerie_vocale,s,1)
Exten => 2,1,Goto(appel_interne,s,1)
Exten => 3,1,Goto(stats_nb_appels,s,1)
Exten => 4,1,Goto(meteo,s,1)

Exten => i,1,Playback(mauvais_choix)
Exten => i,n,Goto(central,10,1)

Exten => t,1,Playback(time_out_depasse)
Exten => t,n,Hangup()


[messagerie_vocale]

[appel_interne]

[stats_nb_appels]

[meteo]
```

2. La messagerie vocale

Dans le fichier voicemail.conf, il faut paramétrer les boites vocales de nos deux softphones en respectant la syntaxe suivante :

mailbox => password,name[,email[,pager_email[,options]]]

Rajoutons donc les deux lignes suivantes dans le contexte [default] :

101 => 111,Liv

102 => 222,Alioune

Le fait de préciser les adresses mail permet d'envoyer un message de notification à l'utilisateur comme quoi il a par exemple un nouveau message dans sa boite vocale. Nous n'avons pas créé de serveur de mail, donc inutile dans notre cas.

```
[messagerie_vocale]
Exten => s,1,Background(accueil_messagerie)
Exten => s,n,WaitExten(15)
Exten => 1,1, VoiceMailMain()
Exten => 1,n, Hangup()
Exten => 2,1,GotoIf($[${CALLERID(num)} = 101]?oui:non)
Exten => 2,n(oui), VoiceMail(101,default)
Exten => 2,n,Hangup()
Exten => 2,n(non), VoiceMail(102,default)
Exten => 2,n,Hangup()
```

3. Les appels internes

```
[appel_interne]
Exten => s,1,Background(accueil_appels_internes)
Exten => s,n,WaitExten(5)
```

```
Exten => 101,1,Macro(blacklist)

Exten => 102,1,Macro(blacklist)

Exten => i,1,Playback(mauvais_choix)

Exten => i,n,Goto(central,10,1)


Exten => t,1,Playback(time_out_depasse)

Exten => t,n, Goto(central,10,1)



[macro-blacklist]

Exten => s,1,GotoIf($[${BLACKLIST()]?blocked,1)

Exten => s,n,GotoIfTime(00:00-23:59|mon-sat|*|*?date_correcte, ${MACRO_EXTEN},1)

Exten => s,n, Playback(date_incorrecte)

Exten => s,n, Goto(central,10,1)

Exten => blocked,1,Playback(tu_es_blackliste_vas_t_en)

Exten => blocked,n,Hangup()



[date_correcte]

Exten => 102,1,Dial(SIP/102,10,m);m pour music on hold

Exten => 102,n,GotoIf($["${DIALSTATUS}" = "BUSY"]?busy:unavail)

Exten => 102,n(unavail),Voicemail(102@default,u)

Exten => 102,n,Hangup()

Exten => 102,n(busy),VoiceMail(102@default,b)

Exten => 102,n,Hangup()
```

Pour ajouter le numéro 101 dans la “blackliste” d’ AstDB, il faut lancer la commande dans le terminal d’Asterisk (CLI ou «command-line interface ») :

```
Asterisk -h
```

CLI*>database put blacklist 101 1

4. Les statistiques d’appels

Dans le contexte [globals], créer une variable courante COUNT

COUNT=0

Dans le contexte [central], incrémenter le nombre d’appels émis :

[central]

Exten => 10,1,Answer()

Exten => 10,2,GotoIf($[${CALLERID(num)} = 101]?oui:non)

Exten => 10,3(oui),Set(COUNT=${DB(nb_appels_101/count)})

Exten => 10,4,Set(COUNT=$[${COUNT} + 1])

Exten => 10,5,Set(DB(nb_appels_101/count)=${COUNT})

Exten => 10,6,Goto(central,10,7)

Exten => 10,3(non),Set(COUNT=${DB(nb_appels_102/count)})

Exten => 10,4,Set(COUNT=$[${COUNT} + 1])

Exten => 10,5,Set(DB(nb_appels_102/count)=${COUNT})

Exten => 10,6,Goto(central,10,7)

Exten => 10,7,Background(accueil_central)

Exten => 10,n,WaitExten(15)

...

Pour ajouter les deux variables dans AstDB, dans le CLI d'Asterisk :

CLI*>database put nb_appels_101 count 0
CLI*>database put nb_appels_102 count 0

Dans le contexte [stats_nb_appels] :

[stats_nb_appels]

Exten => s,1,GotoIf($[${CALLERID(num)} = 101]?oui:non)

Exten => s,n(oui),Set(COUNT=${DB(nb_appels_101/count)})

Exten => s,n,SayNumber(${COUNT})

Exten => s,n,Goto(central,10,1)

Exten => s,n(non),Set(COUNT=${DB(nb_appels_102/count)})

Exten => s,n,SayNumber(${COUNT})

Exten => s,n,Goto(central,10,1)

5. La consultation de données météorologiques.

Il s'agit de l'utilisation de l' « Asterisk Gateway interface » ou AGI, qui fournit une interface standard permettant à des programmes externes de contrôler/interagir avec le plan d'appel. Généralement, les scripts AGI sont utilisés pour communiquer avec des bases de données relationnelles (PostgreSQL ou MySQL) ou bien pour accéder à des ressources extérieures. Dans notre cas, il s'agit d'un script en langage PHP qui permet de télécharger sur Internet un rapport de mesures météorologiques et de les préciser vocalement à l'appelant.

Nous n’allons pas discuter ici du script PHP, c’est en fait un script provenant du livre « Asterisk : The Future of the Telephony ».

Les scripts AGI sont normalement placés dans le répertoire « /var/lib/asterisk/agi-bin ». Il suffit alors de les lancer dans le dialplan avec l’application « AGI(script.agi) »

Dans le dialplan :

[meteo]

Exten => s,1,AGI(script_meteo.agi)

Conclusion

Au terme de ce bureau d'étude, nous avons réussi à mettre en place quelques fonctionnalités permettant une interactivité (basée exclusivement sur de la VoIP à travers le protocole SIP) entre/de nos deux clients par l'intermédiaire de notre serveur Asterisk. Ce qui nous a permis de réaliser que nous avons traité qu'une infime partie de tout ce qui est possible de faire. Il est reconnu que cette solution open-source serait susceptible de transformer l'industrie des télécommunications.

Ce type de système permet par exemple de le connecter à son ancien PBX, permettant ainsi d'élargir le panel de fonctionnalités (fournir un accès aux services de VoIP par exemple). Les systèmes IVR (Interactive Voice Response : exemple avec le script php précédent) dans l'industrie sont très couteux : Asterisk représente alors une excellente solution.

Il est possible d'imaginer toutes formes d'application à la vie courante :

- Surveiller les enfants en contrôlant les téléphones de la maison (en mode écoute) (- :
- Contrôler les appels téléphoniques : en vacances toutes les lignes sont coupées sauf les appels d'urgence (si les voisins ont les clefs par exemple)
- Contrôler le système d'alarme avec son téléphone (permettant de le désactiver si « Grand-Maman » arrive pour récupéer du sucre alors que l'on est en voyage) (- :

Le nombre d'applications possibles est vraiment impressionnant : la mise en place du serveur « Festival » (permettant de transformer du texte en voix), par exemple, faisait partie de nos objectifs.

En conclusion, nous pouvons dire que ce bureau d'étude nous a permis de comprendre et d'avoir un aperçu des nombreuses possibilités offertes par Asterisk.

Principales sources :

- Asterisk The Future of Telephony (O'Reilly)

- http://www.voip.org
- http://www.asteriskguru.com