# 12

# Asterisk

Les grandes entreprises sont dotées de centraux téléphoniques, appelés autocommutateurs, PABX (Private Automatic Branch eXchange) ou plus simplement PBX.

Un PBX est une entité logique, presque toujours gérée par un équipement matériel physique dont la fonction est au moins triple : router les appels au sein d'un réseau privé, interconnecter les réseaux et gérer les services de téléphonie.

Ce chapitre se penche sur un PBX d'un genre nouveau, le logiciel Asterisk, qui remplit les mêmes fonctions qu'un PBX professionnel de haut niveau. Aucun équipement spécifique n'est nécessaire, et il suffit d'installer le logiciel sur un ordinateur, librement et gratuitement. Ce type de logiciel est appelé IPBX, ou PBX-IP.

Grâce à son architecture modulaire, à sa facilité de mise en œuvre rapide et à son fonctionnement simplifié, Asterisk peut même être installé par des particuliers, qui peuvent ainsi exploiter les ressources gigantesques dont il dispose.

## Introduction aux PBX

Seul élément du réseau à connaître la localisation de chaque terminal téléphonique, le PBX a pour fonction principale le routage. Les terminaux sont des entités élémentaires, ce qui réduit leur coût unitaire et permet leur gestion centralisée.

Lorsqu'on ajoute un terminal téléphonique au sein d'une entreprise, il n'est pas nécessaire de modifier les autres terminaux pour les en informer. C'est le PBX qui centralise l'intelligence du réseau et effectue les tâches de connectivité, de mise en relation des interlocuteurs et de gestion des communications locales au réseau. Il assure en outre la liaison avec le réseau téléphonique commuté global. Autrement dit, le PBX fait office de passerelle téléphonique pour les communications locales (d'un point de vue logique et

non physique), mais aussi pour celles entre les utilisateurs du réseau local et les utilisateurs reliés au réseau téléphonique traditionnel.

De cette façon, les communications locales ne sont plus facturées par l'opérateur de téléphonie, puisqu'elles n'arrivent pas jusqu'à lui. Gérées en interne par le PBX, elles deviennent gratuites. De plus, les services mis en place sur le PBX (annuaire, messagerie, etc.) sont indépendants de l'opérateur téléphonique.

Un autre avantage des PBX est que l'entreprise n'a plus besoin d'avoir autant de lignes téléphoniques extérieures qu'elle dispose de lignes en interne. Dans la mesure où il est peu probable que tous les téléphones d'une entreprise soient utilisés en même temps, il est possible de limiter le nombre de lignes extérieures à ouvrir en majorant statistiquement l'usage qui est fait des téléphones. Ainsi, toutes les lignes internes peuvent communiquer, à condition qu'elles ne le fassent pas toutes en même temps. C'est ce qu'illustre la figure 12.1.

Dans la pratique, les utilisateurs ne perçoivent pas cette limitation, à de rares exceptions près, comme lors d'incendies, au cours desquels certains appels peuvent ne pas aboutir en raison de l'affluence des communications. C'est la raison pour laquelle on recommande de réduire au minimum les appels dans ces circonstances particulières.

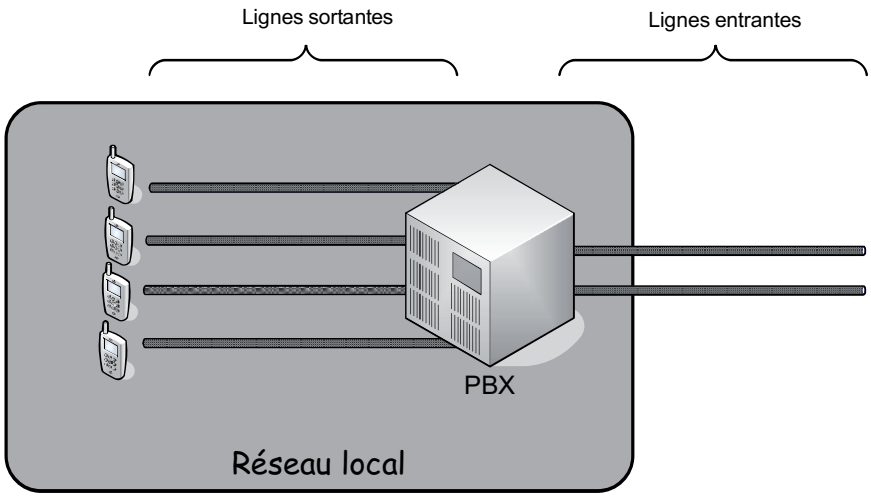



**Figure 12.1**

*PBX pour la gestion des appels*

Les PBX peuvent servir à héberger différents services téléphoniques, comme un serveur vocal, un répondeur personnalisé, la redirection d'appels ou encore la tenue de journaux d'appels. Nous donnons un peu plus loin une liste de quelques services parmi les plus connus.

Les PBX assument enfin la fonction d'interconnexion de réseaux de nature différente. En effet, il ne suffit pas qu'une entreprise souhaitant passer à la ToIP s'équipe de terminaux téléphoniques IP. Pour permettre aux utilisateurs de joindre le réseau téléphonique RTC, une passerelle doit être hébergée au sein du PBX. Enjeu majeur, c'est l'interconnexion entre réseaux qui permet de fédérer les réseaux et d'offrir une plate-forme de services transparente pour l'utilisateur, lui permettant d'accéder aux mêmes services, quel que soit le réseau qu'il utilise.

## Présentation d'Asterisk

Dans la plupart des langages informatiques, l'astérisque (dont le symbole est *) est utilisé comme caractère générique (en anglais *wildcard),* pour remplacer n'importe quel autre caractère ou ensemble de caractères. Il s'agit en quelque sorte d'un joker, la carte ou valeur qui remplace toutes les autres. C'est de ce concept de généricité, évoquant à la fois la souplesse, l'adaptabilité et la puissance, que tire son nom le logiciel Asterisk.

Asterisk est un PBX-IP, ou IP PBX ou encore IPBX. Complet et performant, il offre une plate-forme personnalisable et modulable pour la mise en œuvre de services de téléphonie. Il garantit une très large interconnexion avec plusieurs serveurs PBX, mais aussi avec des réseaux de téléphonie non-IP.

Développé en 2001 par Mark Spencer, de la société américaine Digium, il continue d'être fortement soutenu par cette dernière. En France, c'est la société Eikonex qui assure le catalogue commercial de Digium et propose des formations sur le logiciel, ainsi que le développement d'applications pour des centres d'appels. Eikonex commercialise aussi des cartes servant d'interfaces entre différents réseaux et également des ordinateurs complets, préinstallés et équipés de l'ensemble des équipements optionnels dont on souhaite disposer.

Asterisk étant un logiciel libre d'utilisation, ses sources sont téléchargeables sous licence GNU GPL (General Public License). Cela permet à une importante communauté de contribuer à son développement. Des forums libres et actifs enrichissent, testent, mettent à jour et améliorent en permanence le logiciel. Bien qu'initialement conçu pour fonctionner sous Linux, il est aujourd'hui multiplate-forme et s'installe aussi bien sur OpenBSD que FreeBSD, Sun Solaris, MacOS X ou Windows.

L'enjeu d'une offre telle qu'Asterisk est colossal. Pour peu que l'on dispose des connaissances requises, il devient possible de remplacer une lourde et très onéreuse mise en œuvre d'un équipement PBX par un simple ordinateur équipé du logiciel gratuit, éventuellement muni de cartes d'interfaces pour l'interconnexion avec différents types de réseaux non-IP. Le logiciel se pose en rival viable et robuste dans un marché dominé par les géants Alcatel, Nortel, Cisco, 3Com, Avaya ou Siemens, pour ne citer que quelques-uns des équipementiers les plus connus.

Alléchées par une économie d'environ 50 % par rapport aux solutions hardware classiques, les entreprises multiplient l'adoption de ce type de solution.

## Fonctionnalités

Asterisk propose toutes les fonctionnalités d'un standard téléphonique de niveau professionnel, des plus élémentaires aux plus complexes. Non seulement, il permet de gérer le routage des appels au sein du réseau, mais en plus il supporte une large gamme de services, notamment les suivants (pour la liste exhaustive, voir le site de l'éditeur, à l'adresse *http://www.asterisk.org*) :

- Authentification des utilisateurs appelants.
- Serveur vocal, ou standard d'accueil téléphonique automatisé, aussi appelé IVR (Interactive Voice Response). Cette fonction permet de demander à l'appelant le service qu'il souhaite utiliser et d'effectuer le routage correspondant.
- Numérotation abrégée pour définir des raccourcis.
- Transfert d'appel.
- Filtrage des appels.
- Messagerie vocale (répondeur automatique).
- Notification et écoute par e-mail des messages laissés sur son répondeur *(voicemail)*.
- Gestion des conférences.
- Double appel.
- Mise en attente.
- Journalisation des appels.
- Facturation détaillée.
- Enregistrement des appels.

Le logiciel peut être utilisé comme une passerelle ToIP hétérogène. Par exemple, des utilisateurs utilisant différents protocoles de signalisation, comme H.323 ou SIP, peuvent être mis en relation. C'est le logiciel qui se charge d'effectuer les conversions de signalisation. De la même manière, il peut servir de passerelle pour joindre des correspondants dans le réseau téléphonique RTC. Enfin, le logiciel est modulable et extensible au moyen de scripts et de modules implémentés en langage C ou Perl.

## Compatibilité

Les supports protocolaires d'Asterisk sont très larges. La signalisation sur IP est pleinement supportée avec les protocoles standardisés les plus courants, notamment H.323, SIP et MGCP, mais aussi avec les protocoles IAX (Inter Asterisk eXchange), conçu dans le cadre du projet Asterisk, et SCCP (Cisco Skinny), conçu par Cisco. L'interopérabilité est également assurée vers la téléphonie standard RTC (support pour E&M, E&M Wink, FXS, FXO, GR-303, Loopstart, Groundstart, Kewlstart, MF and DTMF, Robbed-bit Signaling (RBS) et MFC-R2), ainsi que vers la téléphonie RNIS (support pour 4ESS, BRI (ISDN4Linux), DMS100, EuroISDN, Lucent 5E, National ISDN2 et NFAS).

Par ailleurs, le logiciel supporte notamment les codecs audio suivants : G.711 (compatible avec la loi A et la loi ?), ADPCM, G.723.1, G.726, G.729 (soumis à une licence de la

société Digium), GSM, iLBC, Linear, LPC-10 et Speex. Au niveau vidéo, le support des codecs de référence H.263 et H.263+ est aussi fourni. Ce support des codecs les plus réputés offre une large palette de possibilités et couvre l'essentiel des besoins.

## Cible et usage

La première vocation d'Asterisk est de remplacer les PBX d'entreprise, très coûteux, et dont les configurations diffèrent d'un équipement à l'autre. L'objectif est de proposer un logiciel capable de rivaliser avec ces équipements professionnels, à commencer par le support des fonctionnalités de localisation et de mise en relation des utilisateurs.

S'il est naturel de concevoir la présence d'un central téléphonique dans un cadre professionnel, l'exploitation d'une telle puissance fonctionnelle peut-elle s'avérer judicieuse dans un cadre domestique ? Dans un cadre privé, qui souhaiterait disposer d'un mécanisme de standard automatique ? Qui souhaiterait demander à l'appelant de saisir une touche correspondant au membre de la famille qu'il cherche à joindre ?

Un PBX paraît *a priori* trop puissant pour la maison, voire incongru. En réalité, un PBX est bien plus que cela, et certaines de ses fonctionnalités étonnantes et souvent méconnues sont parfaitement adaptées aux besoins des particuliers.

### Réduire les coûts en appelant de l'extérieur au tarif domestique

De nombreux FAI proposent désormais la téléphonie gratuite illimitée vers l'étranger. L'abonné, s'il est chez lui, n'a donc pas à payer les appels longue distance et longue durée vers les destinations proposées. En revanche, il est facturé très cher pour les mêmes appels dès s'il se trouve en dehors de chez lui, par exemple s'il appelle à partir de son téléphone portable. Une solution bon marché à ce problème consiste à utiliser Asterisk comme relais.

Le principe du relais est simple. L'abonné dispose d'un serveur Asterisk correctement configuré chez lui. Lorsqu'il est en déplacement, il appelle le serveur Asterisk. Celui-ci lance automatiquement un processus d'authentification (par exemple en demandant à l'appelant de saisir un code d'accès ou en n'acceptant les appels que si le numéro de téléphone de l'appelant est visible et connu du système). Au terme de l'authentification, le serveur demande à l'appelant de saisir le numéro d'appel de la personne qu'il veut joindre et le compose automatiquement. Il agit dès lors comme un relais pour mettre en relation l'appelant et la personne que ce dernier souhaite joindre.

Dans les conditions que l'on a mentionnées, l'appel composé par le serveur Asterisk vers la destination souhaitée est gratuit. En effet, le serveur étant situé chez l'appelant et relié à la ligne du fournisseur d'accès, il bénéficie des tarifs proposés par ce dernier. Au total, seul l'appel vers le serveur Asterisk est facturé.

Les étapes que nous avons mentionnées ne sont pas si contraignantes qu'elles peuvent paraître à première vue. La même démarche est exploitée par les opérateurs de téléphonie dits alternatifs, qui vendent des cartes prépayées « à gratter » : en téléphonant au numéro indiqué, on s'identifie avec le code gratté avant d'appeler son correspondant.

La seule réelle contrainte est que le serveur Asterisk doit gérer deux appels en parallèle : celui qu'il reçoit de l'appelant et celui qu'il émet pour le compte de ce dernier vers le destinataire. Cela implique de disposer de deux lignes téléphoniques distinctes. Nous verrons que plusieurs opérateurs proposent à moindre coût des lignes téléphoniques IP associées à un même numéro de téléphone.

Pour aller plus loin dans cet exemple, on peut optimiser encore le fonctionnement décrit précédemment grâce à la procédure dite de call-back. Dans ce scénario, l'appel initial est inversé, c'est-à-dire remplacé par un appel initié par le serveur Asterisk :

1. L'appelant appelle son serveur.
2. Contrairement au cas précédent, ce dernier ne répond pas automatiquement à l'appel, mais repère le numéro de l'appelant et attend que ce dernier raccroche. Si ce numéro est reconnu comme étant habilité à utiliser le service de call-back, le serveur Asterisk lance la procédure de call-back en initiant un appel vers l'appelant.
3. Une fois en contact avec l'appelant, le serveur l'invite à saisir ses commandes et à indiquer le numéro de la personne à contacter.
4. Le serveur Asterisk appelle le correspondant.

Là encore, deux lignes téléphoniques sont nécessaires pour faire fonctionner le service. Si l'on suppose que ces lignes sont forfaitaires, un coût mensuel, parfois inclus dans l'abonnement Internet, permettant d'appeler vers plusieurs destinations, l'appel est gratuit. Le call-back permet ainsi de bénéficier de tarifs généralement avantageux.

La figure 12.2 illustre la procédure de call-back.

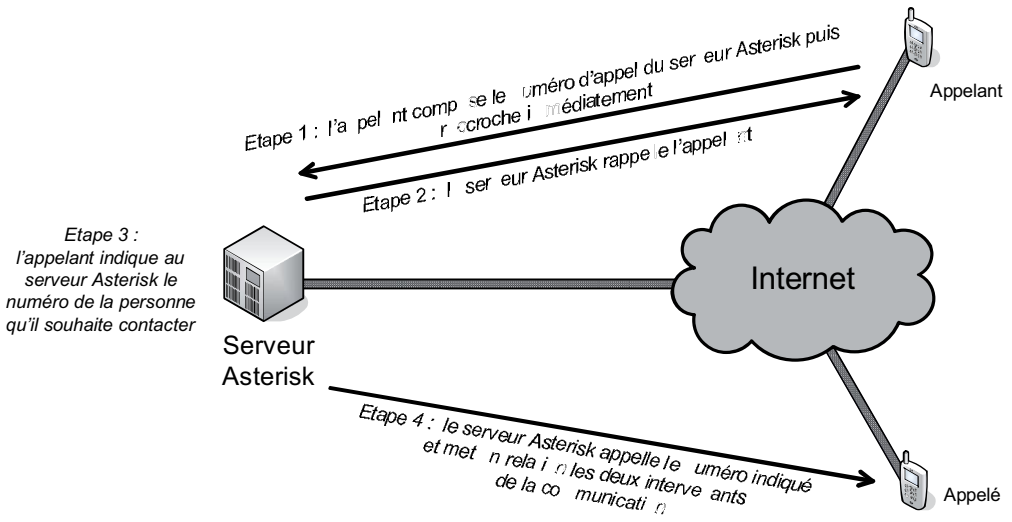



**Figure 12.2**

*Procédure de call-back*

### Assurer le nomadisme des utilisateurs

Plusieurs services permettent de disposer d'un numéro de téléphone IP. Par exemple, l'option SkypeIn de Skype fournit un numéro géographique aux utilisateurs. Étant au format international, ce numéro est utilisable depuis n'importe quelle ligne téléphonique. L'utilisateur est de la sorte joignable quel que soit l'endroit où il se trouve, pour peu qu'il dispose d'une connexion à Internet.

Avec Asterisk, si un utilisateur voyage à l'étranger, il lui suffit de configurer son serveur (relié à sa ligne téléphonique habituelle) pour demander une redirection des appels entrants vers un poste IP spécifique. Dès qu'un appelant souhaite joindre cet utilisateur, il lui suffit de composer son numéro de téléphone pour que l'appel aboutisse au serveur Asterisk, lequel n'a plus qu'à activer la règle lui permettant de relayer l'appel vers le poste IP.

Le même service de nomadisme est disponible avec un téléphone portable en activant l'option d'appels internationaux, si ce n'est que c'est l'appelé qui est généralement facturé pour la redirection d'appel de son pays d'origine vers le pays où il se trouve.

### Améliorer les services téléphoniques

Avec le serveur Asterisk, tous les services téléphoniques usuels sont disponibles. Si certains d'entre eux sont plutôt réservés à des usages professionnels, d'autres sont destinés au grand public. Il est notamment possible, et gratuit, de disposer d'une numérotation abrégée, d'interroger sa messagerie à distance, de recevoir des messages téléphoniques sur sa messagerie électronique en format audio, d'avoir le détail complet des communications passées, ou encore, sous réserve de disposer de l'accord des participants, d'enregistrer des conversations, de lancer des conférences audio, etc.

## Installation de base

En exploitant un ensemble de fichiers restreint, le PBX Asterisk offre une gamme de possibilités de configuration extrêmement large, répondant aux besoins les plus divers et permettant une personnalisation très évoluée.

Au premier abord, cette richesse d'exploitation et ce haut niveau de personnalisation et d'optimisation peuvent sembler rebutants pour les débutants. Plus encore, si la documentation d'Asterisk est globalement assez riche, elle est surtout disponible en langue anglaise. Cela vaut aussi pour les nombreux forums de discussion et d'aide dédiés.

L'ambition de cette section est de fournir une approche suffisamment simple pour initier le lecteur aux concepts fondamentaux d'Asterisk et lui permettre de disposer en un temps record d'une première version élémentaire, mais fonctionnelle, du serveur.

Sans prétendre à l'exhaustivité, puisqu'un livre entier serait nécessaire pour exposer en détail toutes les possibilités et options disponibles, nous nous contenterons d'une approche synthétique de ces concepts nourrie de nombreux exemples.

Nous commencerons par installer et exécuter le serveur avec une configuration par défaut. À ce stade, le serveur sera fonctionnel sans être exploitable. À partir de cet environnement, stable et propre, et auquel il sera possible de revenir ultérieurement en cas de problème, nous passerons à l'étape de configuration des fichiers, avant de nous pencher sur la mise en place de quelques services de base. Nous évoquerons ensuite des techniques d'optimisation de la configuration. Nous terminerons enfin par la présentation de fonctionnalités plus évoluées et fournirons des pistes pour poursuivre l'étude du logiciel de différentes manières.

## Mise en œuvre de la plate-forme

Au lieu de se lancer dans la configuration du logiciel, nous allons d'abord utiliser une configuration élémentaire préinstallée dont nous détaillerons le fonctionnement avant de le personnaliser selon nos besoins. De cette manière, nous limiterons les risques de mauvaises manipulations de fichiers et les fonctionnements inattendus. Les modifications et l'enrichissement des paramètres seront effectués de manière progressive sur une base fonctionnelle.

L'architecture d'Asterisk est modulaire. Autour du serveur, qui constitue son cœur et la seule brique indispensable, viennent se greffer plusieurs composants additionnels qui enrichissent ses fonctionnalités. Il est donc possible de n'installer que les éléments dont nous avons besoin, en choisissant certains modules plutôt que d'autres et en laissant la possibilité d'en ajouter ensuite, voire d'en concevoir soi-même.

Rien n'étant imposé, nous pouvons déterminer presque à la carte les compléments à apporter. Par exemple, nous choisirons d'installer ou non un logiciel de synthèse vocale en français, des pilotes supplémentaires pour la gestion de composants hardware, un logiciel de facturation, etc.

Les sections qui suivent présentent quelques composants classiques, dont nous détaillerons les implémentations et exemples sous une plate-forme Linux.

### Se connecter au réseau téléphonique traditionnel

En ToIP pure, le logiciel Asterisk est d'emblée fonctionnel, si bien qu'aucun composant supplémentaire n'est nécessaire. Cependant, le réseau téléphonique traditionnel utilisant une commutation par circuits, et non par paquets comme dans la téléphonie IP, il est nécessaire de s'équiper d'une interface permettant d'effectuer la conversion des signaux d'un flux IP en un flux RTC et réciproquement.

Ces interfaces sont disponibles facilement dans le commerce, généralement sous forme de cartes PCI, à installer sur son PC, ou de boîtiers, à connecter en USB.

Ils offrent plusieurs branchements pour disposer de plusieurs lignes téléphoniques en parallèle.

Selon les modèles, il faut en outre installer le pilote de la carte, fourni par le constructeur, permettant au système d'exploitation de la reconnaître comme nouveau périphérique matériel et de la rendre disponible et configurable avec Asterisk.

La société Digium, qui maintient le logiciel Asterisk, assure la vente de ces composants à des tarifs compétitifs. D'autres sociétés proposent des cartes analogues, mais leur compatibilité avec les composants vendus pas Digium n'est pas toujours garantie.

On distingue les deux formats d'interface suivants :

- **FXS (Foreign eXchange Subscriber)**, qui permet le branchement d'un téléphone analogique sur le serveur Asterisk. Si l'on ne souhaite pas investir dans l'achat d'équipements purement IP, il est possible d'utiliser son téléphone analogique habituel en le reliant au serveur Asterisk par le biais d'une carte FXS. La carte assure la fonctionnalité de conversion dans les deux sens d'un signal numérique en un signal analogique.
- **FXO (Foreign eXchange Office)**, qui permet le branchement du serveur Asterisk sur une ligne téléphonique classique. Pour interagir ave le monde RTC et dépasser le cadre du réseau purement IP, cette carte assure la jonction avec la téléphonie RTC. Elle joue le rôle de passerelle en faisant communiquer tout utilisateur connecté à Asterisk avec des utilisateurs connectés au réseau RTC.

Dans le cas où l'on souhaite supporter la connectivité avec le réseau téléphonique RTC, il est nécessaire d'installer les drivers Zaptel et Libpri. Tous deux concernent la gestion des cartes FXO/FXS avec Asterisk pour des composants de type Zaptel ou « Zaptel compliant », c'est-à-dire conformes aux composants de type Zaptel, sans être pour autant standards.

**Attention !**

Il est indispensable d'installer ces modules avant l'installation du serveur Asterisk proprement dit. En effet, ces pilotes étant dépendants de composants matériels, le serveur doit les prendre en compte lors de sa compilation et ne peut les intégrer par la suite. Si le serveur Asterisk a déjà été installé, il est nécessaire, après avoir compilé et installé les pilotes Zaptel et Libpri, de revenir à l'étape de compilation du serveur. Cette dernière peut alors prendre en compte le chargement des pilotes.

### Télécharger les composants utiles

Les composants d'Asterisk se présentent sous forme d'archives portant l'extension **.tar.gz** et non de fichiers binaires, qu'il faut compiler puis installer manuellement.

1.Commençons par télécharger la dernière version disponible du logiciel Asterisk, à l'adresse *http://www.asterisk.org/download* (ou *ftp://ftp.digium.com/pub/*). Le site est en anglais, mais la section de téléchargement (download) est facilement identifiable. Comme le logiciel

est libre et assez répandu, il dispose de multiples miroirs, dont un moteur de recherche fournira rapidement les liens. On y trouve les modules récapitulés au tableau 12.1.

**Tableau 12.1 – Principaux modules du logiciel Asterisk**

| Module | Description |
|---|---|
| Asterisk | Cœur du logiciel, ce programme est le seul véritablement indispensable à son fonctionnement. Il est donc indispensable de le télécharger. |
| Asterisk-addons | Ce paquetage comporte le code source du logiciel Asterisk, ainsi que plusieurs modules complémentaires qui peuvent se révéler utiles. Il est vivement recommandé de l'installer. |
| Asterisk-sounds | Ces modules sont fournis sur plusieurs fichiers de paquetage. Ils fournissent une quantité de sons qui peuvent être utilisés dans des messages d'accueil ou pour signaler à l'appelant diverses informations. Ces messages audio sont disponibles en trois langues, anglais, espagnol et français, et sous plusieurs formats de codec, comme G.711, G.722, G.729 et GSM. L'utilisation de ces sons n'étant pas limitative, il est possible, en respectant les formats supportés par Asterisk, d'en ajouter de sa propre composition ou issus d'autres sources. Ces paquetages ne sont pas indispensables, mais seulement pratiques. Ils peuvent être ajoutés ultérieurement. |
| Libiax | Cette bibliothèque de codes source pour les communications utilisant le protocole IAX n'est pas indispensable. Elle est surtout destinée au développement de clients IAX. Nous reviendrons, plus loin dans ce chapitre, sur le protocole IAX. |
| Libpri | Cette bibliothèque est utilisée pour assurer l'interface avec différents types de réseaux non-IP. |
| Zaptel | Ce paquetage contient les pilotes permettant de prendre en charge les cartes d'interface avec les réseaux non-IP. La section qui suit présente les cas où il est indispensable d'installer ce composant. |

**Remarque**

Au lieu de télécharger les sources au format archive, il est possible de télécharger un installeur du logiciel. L'installeur a l'avantage d'être totalement automatisé, et il évite de devoir décompresser puis installer manuellement le logiciel. Néanmoins, il est propre à une distribution particulière, ce qui implique de trouver l'installeur adéquat. Ce dernier se présente le plus souvent comme un fichier portant l'extension **.rpm** pour les distributions Redhat et Mandriva ou **.deb** pour Debian. Les habitués pourront préférer cette méthode à celle que nous présentons, qui reste indépendante de la distribution utilisée.

### Décompresser les sources

Une fois les téléchargements terminés, on peut ouvrir un terminal, se placer à l'emplacement où les archives ont été téléchargées puis saisir la commande qui suit pour chacune des archives que l'on a récupérées :

```
tar -xzvf  nom_du_composant_à_installer
```

*nom_du_composant_a_installer* représente le nom de l'archive téléchargée. Chacune des archives a été décompressée dans un répertoire portant le nom du composant. Les sons n'ont pas besoin d'être installés ; il suffit, une fois l'installation effectuée, de les placer dans le répertoire de sons d'Asterisk (par défaut **/var/lib/asterisk/sounds**). Par contre, les autres programmes, doivent être installés.

### Installer les programmes

Les commandes suivantes permettent d'effectuer la compilation et l'installation d'un composant :

```
cd nom_du_repertoire_du_composant_à_installer
make
make install
```

La première ligne permet de se placer à l'intérieur de répertoire qui a été décompressé précédemment, la seconde lance la compilation du programme et la troisième lance son exécution. Les mêmes commandes sont à reproduire pour chacun des composants, en se plaçant chaque fois dans le répertoire adéquat.

**Important**

Il faut respecter l'ordre d'installation des composants. Si l'on souhaite les utiliser, les modules Libpri et Zaptel doivent impérativement être installés avant le serveur Asterisk. Le module Asterisk-addons peut être installé à la fin.

## Lancement du serveur et exploitation

L'étape d'installation achevée, il faut la tester en lançant le serveur Asterisk, afin de vérifier son bon fonctionnement.

Asterisk n'impose pas de fonctionnement par défaut. Il ne dispose d'ailleurs pas de fichier de configuration préétabli lui conférant initialement un modèle de fonctionnement. La manière dont les appels téléphoniques sont dirigés n'est donc pas encore spécifiée. À ce stade, le serveur n'est pas exploitable.

Pour l'utilisateur débutant, il est préférable de commencer en partant d'une base de fichiers que l'on adaptera par la suite selon ses besoins. Nous allons demander à Asterisk de nous préparer une configuration initiale. Pour cela, nous utilisons un terminal, et nous nous plaçons dans le répertoire source d'Asterisk (celui dans lequel nous avons lancé l'installation), pour saisir la commande suivante :

```
make samples
```

À présent, des fichiers standards sont générés et exploitables.

**Remarque**

Cette commande est utilisable à tout moment pour revenir à un état de base dont la configuration est valide, sans avoir à lancer une nouvelle installation du serveur. C'est une configuration de référence.

Dans cette section, nous n'allons pas modifier les fichiers de configuration, mais simplement lancer le serveur Asterisk. À la section suivante, nous montrerons comment personnaliser les fichiers de configuration afin de gérer des services.

Il existe deux modes différents de lancement d'Asterisk, le mode serveur et le mode client :

- **Mode serveur.** C'est le mode de fonctionnement principal, dans lequel le serveur se met en écoute des clients et prend en charge leur demande de connexion et de communication.
- **Mode client.** Le client Asterisk permet de se brancher au serveur Asterisk et de l'interroger pour lui demander des informations sur son état courant, ou bien pour lui donner de nouvelles directives qui seront prises en compte dynamiquement et modifieront son comportement.

Le mode client nous servira dans les sections suivantes à valider le fait que le serveur est correctement lancé et disponible. Il s'agit donc d'un outil d'administration interactif pratique, sans qu'il soit pour autant indispensable. Le lancement en mode client n'est possible que si une instance d'Asterisk en mode serveur a été préalablement effectuée.

Chacun de ces deux modes est obtenu par le biais de la commande *asterisk*, mais avec des arguments de commande différents dans chaque cas.

### Lancer Asterisk en mode serveur

Le mode serveur d'Asterisk peut être lancé de deux façons, selon que nous souhaitons un lancement automatique (à chaque démarrage du système d'exploitation) ou manuel (uniquement pendant la session courante).

Pour un lancement automatique du serveur, la commande est la suivante :

```
/usr/sbin/safe_asterisk
```

Pour un lancement manuel, la commande devient :

```
asterisk -vvvc
```

La succession des options *v* fournit un niveau de messages informatifs concernant le fonctionnement du serveur de plus en plus élevé. *vvv* est en fait l'acronyme de *very very verbose.* La lettre *c* demande qu'un message (indiquant que la console de contrôle d'Asterisk est bien activée) s'affiche devant chaque ligne. Le message est généralement **CLI>* (pour Command Line Interface).

### Se connecter à Asterisk en mode client

Une fois le serveur Asterisk démarré, nous pouvons vérifier qu'il est opérationnel grâce au client Asterisk. Ce dernier se connecte pour cela au serveur Asterisk afin de récupérer toutes sortes d'informations d'état du serveur et de connexion des utilisateurs.

Il est aussi possible de saisir des commandes d'administration afin de modifier le comportement courant du serveur. Pour exécuter le mode client d'Asterisk (on parle aussi de mode interactif), il suffit d'entrer la commande suivante dans un terminal :

```
asterisk -r
```

**Remarque**

Dès que cette commande est lancée, le client Asterisk se connecte au serveur Asterisk. Ce dernier est à l'écoute de requêtes de paramétrage et de gestion des connexions dont il a la charge. Ces requêtes sont effectuées au moyen d'une interface en ligne de commande, ou CLI (Command Line Interface). Pour connaître la liste des commandes disponibles, il suffit d'entrer la commande *help*.

Pour disposer d'informations plus complètes, la commande *set verbose* permet de récupérer les journaux.

Nous pouvons donc interroger le serveur pour en obtenir différentes informations, dont voici quelques exemples utiles :

- Savoir qui est connecté. Permet de connaître l'ensemble des utilisateurs (ou *peers*) connectés, selon le protocole qu'ils utilisent. Ce ne sont pas des utilisateurs forcément en cours de communication, mais simplement des utilisateurs connectés au serveur Asterisk, et donc joignables. Pour connaître les utilisateurs connectés qui utilisent le protocole SIP, on utilise la commande `sip show peers`, comme ci-dessous :

```
asterisk*CLI> sip show peers
Name/username  Host         Dyn Nat ACL  Port Status
101/101        192.168.1.1  D  5060      OK (20 ms)
103/103        192.168.1.3  D  5060      OK (17 ms)
105/105        192.168.1.5  D  5060      OK (22 ms)
3 sip peers [3 online , 0 offline]
```

De manière analogue, les utilisateurs qui utilisent le protocole IAX peuvent utiliser la commande *iax2 show peers,* comme ci-dessous :

```
asterisk*CLI> iax2 show peers
Name/Username   Host              Mask             Port  Status
202/202         192.168.1.2  (D)  255.255.255.255  4569 Unmonitored
204/204         192.168.1.4  (D)  255.255.255.255  4569 Unmonitored
206/206         192.168.1.6  (D)  255.255.255.255  4569 Unmonitored
207/207         192.168.1.7  (D)  255.255.255.255  4569 Unmonitored
4 iax2 peers [0 online, 0 offline, 4 unmonitored]
```

- Recharger les informations d'un fichier de configuration. Les fichiers de configuration sont en principe chargés au lancement du serveur Asterisk. Il est possible de les recharger dynamiquement pendant que le serveur Asterisk tourne, sans avoir à le redémarrer.

  Pour recharger l'ensemble des fichiers de configuration, il suffit de saisir dans l'interface en ligne de commande la commande *reload*. Plus spécifiquement, il est possible de ne charger que les modifications apportées à un fichier. De manière générale, si le fichier a pour nom **fichier.conf,** il suffit de saisir la commande *fichier reload* pour recharger ce fichier. Par exemple, pour recharger le fichier **extensions.conf,** nous entrons la commande *extensions reload*. De la même manière, la commande *sip reload* permet de recharger le fichier **sip.conf.**

## Configuration

Le serveur Asterisk est à présent opérationnel. À ce stade, il n'assume encore aucune gestion des appels. L'ensemble des paramètres garantissant son mode de fonctionnement, et plus généralement son comportement pour la gestion des appels, se traduit par un ensemble de fichiers de configuration.

Plusieurs interfaces permettent de modifier ces fichiers. Quoique plus conviviales, ces interfaces n'en demeurent pas moins limitatives en comparaison des possibilités offertes en modifiant directement les fichiers de configuration. C'est pourquoi nous préférons présenter ces derniers.

### *Les quatre catégories d'éléments d'Asterisk*

Nous présentons ici les notions fondamentales du fonctionnement d'Asterisk. Pour comprendre et maîtriser le logiciel, il est indispensable de maîtriser ces notions avant de se lancer dans la configuration.

La configuration du serveur Asterisk comporte les quatre catégories d'éléments suivants :

- **Description des utilisateurs et des terminaux.** Pour être joignable, les utilisateurs (s'ils se connectent en utilisant un programme logiciel de type softphone) comme les terminaux (si les utilisateurs se connectent avec des composants matériels) doivent être identifiables. Un identifiant, qui peut être quelconque (sous forme de chaînes de caractères avec des chiffres ou des lettres), doit être affecté à chaque utilisateur (ou terminal). Une authentification peut être ajoutée pour s'assurer de l'identité des utilisateurs (ou terminaux) et préserver le système des utilisations frauduleuses. Les utilisateurs (et les terminaux) sont recensés dans des fichiers différents selon le protocole de signalisation qu'ils exploitent (SIP, H.323, IAX, etc.).

- **Plan de numérotation *(ou dial plan).*** Après avoir générer les comptes des utilisateurs, il faut ensuite déterminer de quelle manière ils vont communiquer, c'est-à-dire, d'abord, quel est le numéro de téléphone attribué à chaque terminal, mais aussi comment les appels et les services s'effectuent. La notion de plan de numérotation correspond précisément aux règles de routage des appels qui régissent le fonctionnement du système et permettent la mise en relation les interlocuteurs. Il convient de les programmer et de les analyser en profondeur, car elles concentrent l'intelligence de la plate-forme. En spécifiant la manière de réagir aux appels, le système fournit un premier niveau de service aux utilisateurs : le routage des communications. Un fichier unique définit le plan de numérotation **(extensions.conf),** lequel peut éventuellement appeler d'autres fichiers par inclusion.
- **Description des services supplémentaires.** Les services supplémentaires permettent d'assurer toutes sortes de fonctionnalités, telles que la messagerie téléphonique ou la journalisation des appels. Chaque service est défini dans un fichier spécifique, dont nous donnerons quelques exemples.
- **Description du matériel physique.** Si nous restons dans un réseau purement IP, aucun autre matériel qu'un ordinateur n'est nécessaire. Dès lors que nous souhaitons correspondre avec des utilisateurs du réseau téléphonique commuté ou de tout autre réseau non-IP, nous devons recourir à des équipements spécifiques. La description du matériel permet d'indiquer au serveur Asterisk la nature de ces composants, de façon qu'il en tienne compte. Les fichiers étant différents selon le matériel considéré, le serveur Asterisk doit être recompilé avec le chargement de ces fichiers.

Chacun de ces éléments est décrit par un ou plusieurs fichiers. La séparation de ces fichiers apporte une modularité à la gestion de la plate-forme. Nous décrivons à la section suivante les fichiers utilisés et la manière de les programmer. Puis, dans les sections suivantes, nous détaillerons les catégories d'éléments précédemment mentionnées pour expliquer successivement comment s'effectuent la description des utilisateurs et des terminaux, puis la programmation du plan de numérotation et enfin la mise en place de services complémentaires.

## Organisation des fichiers (fichier asterisk.conf)

Les fichiers d'Asterisk sont répartis dans plusieurs répertoires afin de suivre l'organisation classique des systèmes Linux. Le répertoire contenant les exécutables binaires du serveur Asterisk et ses composants principaux est situé par défaut dans le chemin **/usr/bin/.** Il comporte les commandes principales suivantes : *asterisk, astman, astgenkey, safe_asterisk.*

Pour tous les autres fichiers non binaires, le fichier **asterisk.conf** offre une grande souplesse d'utilisation et laisse l'administrateur libre de modifier sa configuration par défaut, en spécifiant l'emplacement dans l'arborescence du système de fichiers utilisé.

En l'absence de ce fichier, les chemins par défaut sont considérés, comme dans l'extrait ci-dessous :

```
[directories]
astetcdir => /etc/asterisk
astmoddir => /usr/lib/asterisk/modules
astvarlibdir => /var/lib/asterisk
astagidir => /var/lib/asterisk/agi-bin
astspooldir => /var/spool/asterisk
astrundir => /var/run/asterisk
astlogdir => /var/log/asterisk
```

Le tableau 12.2 récapitule les descriptions correspondant aux différentes directives utilisables dans le fichier **asterisk.conf.**

**Tableau 12.2 – Directives indiquant l'emplacement des répertoires d'Asterisk**

| Directive | Description |
|---|---|
| *astetcdir* | Spécifie le répertoire contenant l'ensemble des fichiers de configuration (portant l'extension **.conf**). |
| *astmoddir* | Spécifie le répertoire contenant l'ensemble des modules (portant l'extension **.so**). |
| *astvarlibdir* | Spécifie le répertoire contenant l'ensemble des bibliothèques exploitables, fournissant notamment une galerie de fichiers audio qui peuvent être utilisés dans des services avec des annonces automatisées du serveur (mise en attente, messagerie, annonces d'accueil, etc.) et seront appelés par leur nom dans la programmation de ces services. |
| *astagidir* | Spécifie le répertoire contenant l'ensemble des scripts AGI (Asterisk Gateway Interface), fournissant des fonctionnalités complémentaires implémentées avec un langage de programmation tel que Shell, C, PHP, Perl ou Pascal. |
| *astspooldir* | Spécifie le répertoire contenant l'ensemble des enregistrements audio qui sont réalisés par le serveur Asterisk, notamment pour la messagerie vocale, mais aussi pour l'enregistrement de communications, de conférences, etc. |
| *astrundir* | Spécifie le répertoire contenant le fichier de PID (Process ID) d'Asterisk identifiant de manière unique le processus en charge de la gestion du serveur, qu'il est possible de suivre parmi les autres processus. |
| *astlogdir* | Spécifie le répertoire contenant l'ensemble des fichiers journaux sauvegardant les événements survenus et le traitement qui leur a été appliqué. Il est important de vérifier que ce répertoire ne devienne pas rapidement trop volumineux pour surcharger les capacités du serveur. |

À la suite de la section *[directories],* on trouve une section *[files]* qu'il est vivement recommandé de ne pas modifier. Elle comporte des informations de sécurité, en particulier les droits restreignant l'accès au serveur Asterisk.

La section suivante détaille les principaux fichiers de configuration générale qu'il convient de connaître. De manière générale, un commentaire peut être inséré dans ces fichiers de configuration en le faisant précéder d'un point-virgule.

**Attention !**

Les fichiers ne sont pas interprétés dynamiquement. Autrement dit, pour qu'une modification effectuée dans un fichier soit prise en compte par le serveur en cours de fonctionnement, il est nécessaire de lui imposer un rechargement de ces fichiers par la commande *reload* (introduite à la section présentant la connexion en mode client à Asterisk).

## *Première étape de configuration : Description des utilisateurs et des terminaux (fichiers sip.conf, iax.conf, mgcp.conf, h323.conf, skinny.conf)*

La première étape de configuration du serveur Asterisk correspond à la définition des comptes d'utilisateurs et des terminaux. Ceux-ci sont identifiés par le protocole de signalisation qu'ils utilisent. Il existe donc un fichier de configuration par protocole de signalisation supporté.

A ce stade, les numéros de téléphone ne sont pas associés aux comptes des utilisateurs et aux terminaux. Il faudra attendre la programmation du plan de numérotation pour mettre en place ces associations (voir l'exemple numéro 1, à la fin de la section suivante). Cette distinction entre un utilisateur (ou terminal) et son numéro offre une meilleure visibilité en regroupant les attributions de numéros, indépendamment des paramètres des comptes.

On notera que, de la même manière qu'on déclare un compte d'utilisateur lorsque celui-ci se connecte avec un softphone, il est possible de déclarer des téléphones physiques pour autoriser leur connexion dans le parc géré par Asterisk. On procédera pour cela de manière analogue, en déclarant un compte, et les propriétés associées à ce compte, comme on le ferait pour un utilisateur.

Nous détaillons dans les sections suivantes les fichiers **sip.conf** et **iax.conf.**

### Le fichier *sip.conf*

Le fichier **sip.conf** permet de définir tous les utilisateurs SIP. Il est segmenté en sections, dont chacune débute par une étiquette *(*le *label)* entre crochets.

Le label spécial *[general]* permet d'attribuer des valeurs à des paramètres génériques, tels que le port utilisé. Le label *[user_id]* définit chaque compte d'utilisateur.

Voici un exemple de fichier **sip.conf,** dans lequel deux utilisateurs sont déclarés **:**

```
[general]
port=5060

[david]
username=David
secret=s1p@st3r1sk!
callerid="David" <5551>
type=friend
host=dynamic
nat=yes
canreinvite=no
context=internal
disallow=all
allow=ulaw
allow=gsm
allow=h263

[laurent]
type=friend
host=dynamic
context=internal
username=Laurent
nat=yes
canreinvite
secret=h1O2m3E
callerid="Laurent" <5552>
```

La section *[general]* indique le numéro de port utilisé par tous les utilisateurs, soit ici *5060.* Les sections suivantes renseignent les paramètres de deux compte d'identifiant *david* et *laurent.* Les paramètres les plus utilisés pour la définition de ce compte sont récapitulés au tableau 12.6. Comme on le voit dans l'exemple, tous ne sont pas obligatoires (lorsqu'ils sont optionnels, des valeurs par défaut sont pratiqués si les paramètres ne sont pas présents), et l'ordre dans lequel ils sont donnés n'a aucune importance.

**Tableau 12.6 – Paramètres décrivant un compte d'utilisateur**

| Paramètre | Description |
|---|---|
| *username* | Identifiant de l'utilisateur |
| *secret* | Mot de passe associé au compte |
| *type* | Indique le type de compte, et les restrictions associées. On distingue trois types de comptes :<br>– *Friend* : permet d'appeler et d'être appelé (autorise les appels entrants et sortants).<br>– *User* : permet seulement d'être appelé (appels entrants).<br>– *Peer* . permet de définir une liaison entre deux terminaux seulement. |
| *host* | Spécifie une adresse IP à partir de laquelle l'utilisateur peut accéder à son compte. La valeur *dynamic* autorise une adresse IP fournie dynamiquement, par un serveur DHCP notamment. Cette valeur est donc moins restrictive. |
| *callerid* | Nom de l'utilisateur, entre guillemets, suivi de son extension téléphonique, c'est-à-dire de son numéro d'appel (au format de la RFC 822).<br>Attention : le numéro de téléphone mentionné ici ne constitue pas une association du numéro avec l'utilisateur (cette association sera faite ultérieurement). Ce paramètre permet simplement d'identifier l'utilisateur (ou le terminal) lorsqu'il passe des appels. Autrement dit,cette information est utilisée dans les appels sortants uniquement pour indiquer le nom (si le terminal appelé permet d'afficher cette information) et le numéro de téléphone de l'utilisateur. |
| *context* | Spécifie le type de routage à appliquer pour l'utilisateur. Le type de routage correspond à un contexte défini dans le plan de numérotation (fichier **extensions.conf**). Les communications avec cet utilisateur sont donc soumises au contexte du même nom dans le fichier **extensions.conf.** |
| *language* | Spécifie la langue utilisée pour les fichiers audio.<br>Par exemple : *language=fr*. |
| *allow* | Liste les codecs autorisés par l'utilisateur de ce compte.<br>Par exemple, pour autoriser le codec G.711, selon la loi mu, on saisira: *allow=ulaw*.<br>Ce même paramètre est aussi valable pour spécifier les codecs vidéo (par exemple : *allow=h263*)Il est possible d'en mentionner autant que l'on souhaite, en répétant ce paramètre autant de fois. |
| *disallow* | Interdit les codecs qui sont mentionnés à sa suite. Une valeur possible de ce paramètre est *all*. Dans ce cas, aucun codecs ne sera utilisable par l'utilisateur concerné, sauf ceux spécifiés explicitement dans le (ou les) paramètre(s) *allow*. |
| *nat* | Précise si les flux traversant un réseau utilisent la translation d'adresse (NAT). La valeur du paramètre *nat* est *yes* ou *no*.<br>Ce paramètre est souvent indispensable car l'utilisation du nat est classique, même chez les particuliers. |
| *canreinvite* | Ce paramètre peut prendre les valeurs yes ou no. Si sa valeur est fixée à yes, Lorsqu'une communication est en train de s'établir le serveur Asterisk va récupérer des informations (notamment vers qui envoyer les flux) et ils les réémettra dans un nouveau message d'invitation une fois que la communication sera acceptée seulement.<br>Attention, si l'utilisateur se trouve derrière un NAT, il est indispensable de mettre la valeur de ce paramètre à no pour laisser passer les flux multimédia correctement, car le nouveau message d'invitation du serveur Asterisk ne tiendrait pas compte du NAT. |
| *mailbox* | Indique la boite vocale associée à ce compte. Nous détaillons ce paramètre et son utilisation à la section qui traite de la messagerie audio. |
| *dtmfmode* | Ce champ indique le type de tonalité DTMF qui sera appliqué. Les valeurs possibles sont *inband*, *rfc2833*, *info* ou *auto* |

### Le fichier *iax.conf*

Les clients utilisant le protocole de signalisation IAX sont mentionnés dans le fichier **iax.conf.** Son fonctionnement et sa description sont semblables à ceux du fichier **sip.conf.**

Voici un exemple de fichier **iax.conf** définissant un utilisateur :

```
[general]
bindport=4569

[1234]
username=1234
type=friend
host=dynamic
context=internal
callerid="Guy" <1234>
```

Là aussi, comme pour le fichier **sip.conf** et contrairement à ce qu'on pourrait croire, les nombres ne sont pas des affectations de numéros de téléphones à des utilisateurs. Ce ne sont que des identifiants de comptes. Seul le plan de numérotation permet de réaliser la correspondance d'un numéro de téléphone avec un utilisateur.

## *Deuxième étape de configuration : le plan de numérotation (fichier extensions.conf)*

Une fois les comptes des utilisateurs et des terminaux définis, il faut leur attribuer des numéros de téléphone pour qu'ils soient joignables. Il faut aussi déterminer la procédure qui s'enclenchera lors de chaque appel (par exemple, si l'appel sera transmis vers deux postes en même temps, s'il sera redirigé vers une messagerie au bout d'un laps de temps, si l'appel sera journalisé dans une base de données, etc.).

Il faut enfin définir les services spéciaux que l'on souhaite activer (par exemple le service de messagerie, les serveurs vocaux disponibles, les salons de conférence mis en place, etc.). Tout cela s'effectue au moyen d'un unique composant : le plan de numérotation.

Le plan de numérotation, ou *dial plan,* est l'élément central de la configuration du serveur Asterisk. Il définit le comportement du serveur PBX. Maître de cérémonie ou chef d'orchestre, c'est lui qui régit les actions à entreprendre, dans quel ordre et dans quel cas, que ce soit pour un utilisateur donné ou pour l'ensemble des utilisateurs.

Ce plan concentre toute l'intelligence et la logique de fonctionnement du réseau téléphonique. C'est pourquoi il est indispensable d'en maîtriser à la fois la syntaxe et la sémantique. Il est constitué d'un ensemble de règles, dont chacune pose les conditions de

son application, ainsi que, lorsque ces conditions sont réunies, les traitements qui seront appliqués.

Le plan de numérotation est défini dans un fichier unique dont le nom est **extensions.conf.** Sous Linux, et avec une installation standard, ce fichier se trouve généralement dans le répertoire **/etc/asterisk/.**

Le plan de numérotation répond à la question : que doit faire le serveur PBX Asterisk lorsqu'il reçoit le flux téléphonique d'un utilisateur ? La réponse est fournie sous forme de règles qui sont structurées et dont la syntaxe est définie par les quatre éléments suivants :

- le contexte
- l'identifiant d'extension
- la priorité
- l'application

Ces éléments décrivent les critères que les flux doivent vérifier et le traitement qui leur sera appliqué le cas échéant.

Le format général d'un plan de numérotation, dans lequel se combinent ces quatre éléments, est le suivant :

```
[contexte_1]
exten => identifiant_d'extension_1, priorité_1, application_1
exten => identifiant_d'extension_2, priorité_2, application_2
exten => identifiant_d'extension_3, priorité_3, application_3

[contexte_2]
exten => identifiant_d'extension_4, priorité_4, application_4
```

On distingue dans cet exemple deux contextes différents, signalés par *[contexte_1]* et *[contexte_2].* Le mot-clé *exten* est utilisé pour définir une extension. Il est suivi d'une flèche, formée par les symboles = et >.

Dans notre exemple, trois extensions sont définies dans le premier contexte, et une dans le second. Chaque extension comporte un identifiant d'extension *(identifiant_d'extension_i),* un numéro de priorité *(priorité_i)* et une fonction applicative *(application_i).* Chacun de ces critères permet de préciser qui est l'appelant, avec quel service (ou personne) il souhaite être mis en relation et comment effectuer la fourniture de ce service.

Nous pouvons lire la première règle comme suit : « Lorsque l'extension *identifiant_d_extension_1* se présente dans le contexte *contexte_1,* nous exécutons l'action *application_1* avec la priorité *priorité_1* ». Il est nécessaire de maîtriser ces éléments

pour affecter un numéro de téléphone à un utilisateur (ou à un terminal) et plus généralement pour déterminer la manière dont les appels se déroulent. Les sections suivantes précisent le rôle de chacun de ces quatre éléments.

### D'abord le contexte

Le plan de numérotation est organisé en sections appelées *contextes*. Un contexte définit un cadre d'application. Par exemple, si un utilisateur compose le chiffre 5 sur son poste téléphonique, ce numéro est intercepté par le serveur Asterisk, lequel peut interpréter différemment ce numéro selon différentes situations, notamment les suivantes :

- Si l'utilisateur ne consulte aucun service, le chiffre 5 peut correspondre au numéro de téléphone abrégé que le serveur a enregistré pour un de ces contacts.
- Si l'utilisateur a préalablement demandé à effectuer un appel longue distance, le chiffre 5 peut être le premier d'un code d'authentification permettant la mise en relation.
- Si l'utilisateur est en relation avec un service vocal, le chiffre 5 peut correspondre au choix d'un service spécifique parmi ceux proposés.
- Si l'utilisateur est en train de consulter son répondeur hébergé par le serveur Asterisk, le chiffre 5 peut être un code de commande pour demander au serveur de réécouter un message.
- Si l'utilisateur est en cours de communication avec un contact, le chiffre 5 peut demander au serveur Asterisk de lancer l'enregistrement de la conversion en cours.

Le serveur Asterisk ne doit donc pas interpréter de la même façon toutes les saisies effectuées sur un terminal téléphonique, mais distinguer les cas de figure. Dans notre exemple, les contextes correspondent à l'état dans lequel l'appel s'inscrit : l'utilisateur peut être en ligne, non authentifié, en relation avec le service vocal, en relation avec un service de messagerie vocale ou en cours de communication. D'où nos cinq contextes.

Pour savoir quel contexte utiliser lors d'un appel, le plan de numérotation se fonde sur les éléments suivants :

- Identité de l'appelant ou de l'appelé, auquel il est possible d'attribuer des contextes particuliers.
- Actions entreprises par les utilisateurs pendant un appel : la saisie de touches par un utilisateur peut engendrer des branchements conditionnels ou inconditionnels (présentée plus loin avec les structures *goto* et *gotoIf*).

De la même manière, les contextes permettent de gérer des catégories d'utilisateurs. Nous pouvons, par exemple, définir un contexte pour les employés commerciaux autorisés à passer des appels longue distance, et un autre pour les employés ingénieurs uniquement autorisés à passer des appels locaux. Dans un cadre domestique, nous pouvons définir un contexte pour les personnes adultes, dans lequel tous les appels sont possibles, et un autre pour les enfants, dans lequel seuls des numéros d'appel prédéfinis sont possibles.

La puissance du serveur Asterisk réside dans cette capacité de personnalisation, qui peut concerner un utilisateur ou un service spécifique, comme dans les cas suivants :

- Un utilisateur peut disposer de services différents de ceux disponibles pour un autre utilisateur. Par exemple, si le poste d'un utilisateur est indisponible au bout de cinq sonneries, l'appel peut être redirigé vers le poste d'un autre utilisateur, tandis que le même cas de figure pour un autre utilisateur déclenche la messagerie.
- Tous les services étant personnalisables par l'administrateur, un administrateur peut décider que la messagerie des utilisateurs se déclenche au bout de cinq sonneries, alors qu'un autre peut décider qu'elle ne s'active qu'au bout de sept sonneries.

À chaque contexte, une politique de gestion particulière peut être définie sous la forme d'une ou plusieurs extensions. Chaque contexte regroupe un ensemble d'extensions constituées d'un identifiant d'extension, d'une priorité et d'une application.

### Les contextes particuliers

Il existe deux contextes particuliers, appelés *[general]* et *[globals]* et tous deux placés au tout début du fichier **extensions.conf.**

#### [general]

Toujours mentionnée avant les autres, cette section permet de définir des options générales appliquées par le serveur Asterisk au plan de numérotation. Par exemple, on peut indiquer à la suite de la section :

```
clearglobalvars=yes
```

Cela a pour effet d'effacer l'ensemble des variables globales enregistrées par le serveur Asterisk. Au contraire, la valeur *no* a pour effet de conserver aux variables globales des valeurs persistantes à chaque rechargement du serveur, même si elles ne sont plus spécifiées dans les fichiers de configuration.

De même, en mentionnant :

```
static=yes
writeprotect=no
```

il est possible de sauvegarder le plan de numérotation par l'interface de contrôle (CLI) d'Asterisk présentée précédemment.

Il est également vivement recommandé de choisir :

```
autofallthrough=yes
```

Cette valeur, prise par défaut dans les versions récentes d'Asterisk, permet de terminer automatiquement un appel pour lequel tous les traitements mentionnés dans le plan de numérotation ont été effectués. À défaut, si ce paramètre a pour valeur *no,* le serveur attend, une fois tous les traitements effectués, que l'utilisateur effectue une nouvelle saisie.

[globals]

La section *[globals]* permet de définir toutes les variables globales susceptibles d'être utilisées dans la suite du fichier. Nous verrons plus loin comment définir de telles variables.

### Ensuite, l'identifiant d'extension

Au sein de chaque contexte, des règles permettent de définir le comportement à adopter, autrement dit la manière dont le routage et le service doivent être rendus.

L'identifiant d'extension définit la destination d'un appel. En première approximation, c'est le numéro d'un poste appelé. Nous verrons que cela peut aussi être un service déterminé ou un choix dans un menu vocal. Il correspond donc à un numéro (ou à un nom) d'appel, attribué indifféremment à une personne physique ou à un service automatisé.

À titre d'exemple, le tableau 12.3 donne quelques identifiants d'extension avec leur assignation respective.

**Tableau 12.3 – Exemples d'identifiants d'extension**

| Identifiant d'extension | Description |
|---|---|
| 0 | Accueil |
| 5 | Répondeur téléphonique |
| 101 | Poste téléphonique d'Albert |
| 102 | Poste téléphonique de Béatrice |
| 103 | Poste téléphonique de Caroline |

Le choix des identifiants d'extension est laissé à la convenance de l'administrateur. Les identifiants peuvent être formés de chiffres (comme pour un numéro de téléphone classique) ou de lettres (comme le nom d'une personne ou d'un service).

**Remarque**

Les espaces sont prohibées. En cas d'utilisation d'espaces, le comportement du système devient imprévisible, même si aucune erreur n'est retournée.

Les utilisateurs peuvent être identifiés par leur adresse e-mail. Si l'identifiant d'extension comporte des lettres, cela implique que le terminal appelant dispose du matériel adéquat, de type clavier, pour saisir l'identifiant d'appel. On peut toutefois recourir à d'autres mécanismes, comme la reconnaissance vocale.

Une règle ne peut être atteinte que lorsque l'exécution de l'action relative a la règle précédente est terminée. Les raisons à cela sont, d'une part, que deux actions peuvent entrer en concurrence (diffusion d'un message sonore avec deux actions, dont la lecture conjointe rendrait les deux messages inaudibles), d'autre part, qu'une règle peut entraîner des sorties conditionnelles ou des sauts vers une autre règle selon un déroulement particulier (par exemple, la saisie d'un choix par l'appelant peut entraîner une autre action que l'absence de choix).

### Les extensions particulières

Asterisk définit quelques extensions particulières permettant non pas de joindre un utilisateur ou un service, mais de fournir un comportement explicite pour des situations particulières. Les trois extensions prédéfinies les plus courantes sont *s, t* et *i.*

#### L'extension *s*

Lorsqu'un flux téléphonique parvient au serveur Asterisk, celui-ci peut lui appliquer un traitement indifférent de la personne ou du service que le flux tente de contacter. Pour cela, l'administrateur utilise l'extension *s* (pour le mot-clé *start*), qui indique que tous les flux (dans le contexte concerné) seront traités par la règle mentionnée.

En voici un exemple :

```
exten => s, 1, Playback (msg_attente)
```

Tout appel soumis a cette règle lance un message audio nommé *msg_attente* pour indiquer à l'appelant que la communication est prise en compte et mise en attente.

#### L'extension *t*

Lorsqu'un certain délai (par défaut 10 secondes) s'écoule sans qu'une extension ait été saisie par l'utilisateur, l'extension *t* (pour le mot-clé *timeout*) est automatiquement appelée par le système.

En voici un exemple :

```
exten => t, 1, Playback (msg_delai_depassé)
```

Si un délai *t* (par défaut 10 secondes) s'écoule, un message nommé *msg_delai_depassé* est automatiquement lancé pour avertir l'utilisateur de l'attente d'une saisie par le système.

### L'extension *i*

Lorsque l'utilisateur saisit une extension qui n'est pas référencée dans le plan de numérotation, l'extension *i* (pour le mot-clé *invalid*) est automatiquement appelée par le système.

En voici un exemple :

```
exten => i, 1, Playback (msg_invalide)
```

Si l'utilisateur saisit une extension inconnue, un message nommé *msg_invalide* est automatiquement lancé pour l'avertir de l'invalidité de sa saisie.

## Les filtres d'extension

Il est possible de définir des identifiants d'extension formés d'un filtre, ou *pattern,* qui représente une syntaxe générique d'identifiant. Cela permet d'offrir un service générique à des groupes d'utilisateurs ou des services spécifiques. En particulier, cela permet de distinguer les appels locaux des appels internationaux en fonction des préfixes de numérotation.

Tout filtre d'extension est précédé d'un caractère de soulignement *(underscore)*. Les caractères spéciaux permettant de définir un filtre sont définis comme indiqué au tableau 12.4.

**Tableau 12.4 – Filtres d'extension**

| Filtre | Description |
|---|---|
| *chaîne_quelconque* | Impose la présence de la chaîne *chaîne_quelconque* dans l'identifiant d'extension. |
| *[caractères_quelconques]* | Remplace un caractère dans un identifiant d'extension parmi l'un de ceux mentionnés entre les crochets. |
| *X* | Remplace un chiffre entre 0 et 9 dans un identifiant d'extension. |
| *Z* | Remplace un chiffre entre 1 et 9 dans un identifiant d'extension. |
| *N* | Remplace un chiffre entre 2 et 9 dans un identifiant d'extension. |
| . | Remplace n'importe quel caractère ou série de caractères. C'est donc un caractère joker qui ne devrait être indiqué qu'avec un filtre suffisamment descriptif. |

La position des caractères spéciaux doit être respectée pour correspondre à l'identifiant d'extension filtré.

L'exemple suivant :

```
_0142XXXXXX
```

s'applique à n'importe quelle extension commençant par *0142* et ayant une longueur de 10 chiffres.

L'exemple suivant :

```
_0Z[12589]XXX
```

s'applique à une extension ayant pour premier caractère le chiffre *0,* pour deuxième caractère un chiffre entre *1* et *9,* pour troisième caractère un chiffre parmi les valeurs *1, 2, 5, 8* ou *9,* puis, pour les trois caractères suivants (les trois symboles *X*), une valeur quelconque entre *0* et *9* et enfin pour le septième caractère (le symbole point) un ou plusieurs chiffres quelconques. Au total, le numéro fait un minimum de sept chiffres, le maximum n'étant pas mentionné.

Le filtre _. (underscore suivi d'un point) remplace n'importe quel caractère ou série de caractères, autrement dit il s'applique à toutes les extensions. Ce filtre d'extension ne devrait donc jamais être utilisé, puisqu'il est toujours vérifié et s'applique sans restriction à tous les appels.

### Les extensions conditionnelles

Une extension définit en principe le numéro d'une personne ou d'un service à joindre, mais elle peut être conditionnée par l'identité de la personne appelante. En effet, un même numéro d'appel peut donner lieu à des traitements différents. Par exemple, dans une entreprise, un unique numéro d'appel pour un secrétariat peut être affecté à deux secrétaires ayant chacune leur spécialisation. Selon la personne appelant le secrétariat, une redirection vers le poste le plus adéquat peut s'effectuer. Il suffit pour cela de mentionner le numéro d'appel de la personne qui appelle juste après l'extension du service à joindre suivi d'un caractère slash.

Par exemple :

```
exten => 101/105, …
```

spécifie une règle concernant l'appelant ayant pour identifiant *105* et appelant le poste *101*.

Les extensions conditionnelles peuvent utiliser des filtres, à la fois pour les identifiants de l'appelé et ceux de l'appelant.

### Puis, la priorité

La priorité définit l'ordre dans lequel la règle doit s'appliquer. Certains traitements nécessitent plusieurs actions à entreprendre pour obtenir le comportement désiré. Par exemple, si un correspondant ne répond pas, l'appelant peut être redirigé vers la messagerie de son correspondant. Dans ce cas, au moins deux étapes correspondent à un même appel : la tentative d'appel vers le correspondant, puis la redirection vers la messagerie.

Il faut ordonner chacune de ces étapes en mentionnant dans le plan de numérotation la priorité affectée à chacune des règles.

La priorité est spécifiée par une valeur numérique entière débutant par la valeur *1*. Plus la valeur attribuée à une règle est faible, plus la priorité qui lui est concédée est importante. Les règles se succèdent tour à tour avec un même identifiant d'extension, selon un ordre croissant.

**Attention !**

Si une priorité est oubliée (par exemple si l'on a une priorité 1 puis une priorité 3, et donc qu'on a omis la priorité 2), le serveur ne peut poursuivre (donc passer à la priorité 3 dans notre exemple). Il importe de veiller à ce que l'ordre affecté à chaque règle n'omette pas une valeur. Lors d'une mise à jour, en particulier, il ne faut pas effacer une règle sans vérifier la cohérence et l'enchaînement avec les priorités qui suivent.

### La priorité *n*

Il est possible d'optimiser la gestion des priorités en se fiant à l'emplacement d'écriture des règles pour en fixer l'ordre.

La notion de priorité numérique présente une contrainte délicate dans le maintien et la mise à jour des règles. Si nous souhaitons ajouter une nouvelle règle entre deux règles, il est nécessaire de modifier explicitement toutes les priorités des règles qui suivent la règle ajoutée.

Considérons la succession de règles suivantes :

```
exten => 98765, 1, action_1
exten => 98765, 2, action_2
exten => 98765, 3, action_3
```

Les applications spécifiées *action_i* sont quelconques. Après la première règle, nous souhaitons ajouter une nouvelle règle exécutant l'action *action_nouvelle*. Nous modifions comme suit la séquence précédente :

```
exten => 98765, 1, action_1
exten => 98765, 2, action_nouvelle
exten => 98765, 3, action_2
exten => 98765, 4, action_3
```

Cette manière de procéder est contraignante en ce qu'elle impose chaque fois de vérifier la cohérence des ordonnancements. Asterisk propose une manière plus simple de gérer les priorités grâce à la priorité *n*. Le *n* spécifie l'action suivante (en anglais *next*). Avec une priorité *n*, les étapes sont suivies selon leur ordre d'apparition dans la séquence du plan de numérotation.

L'exemple précédent initial devient ainsi :

```
exten => 98765, 1, action_1
exten => 98765, n, action_2
exten => 98765, n, action_3
```

L'ajout de la nouvelle règle donne simplement :

```
exten => 98765, 1, action_1
exten => 98765, n, action_nouvelle
exten => 98765, n, action_2
exten => 98765, n, action_3
```

Comme nous le constatons, il n'est pas nécessaire de modifier les règles suivant l'ajout d'une nouvelle action. L'enchaînement des règles est géré automatiquement.

### Enfin, l'application

L'application définit l'action à entreprendre pour appliquer le service sollicité par l'utilisateur appelant. Le serveur Asterisk dispose d'un grand nombre de procédures déterminant le comportement à adopter, c'est-à-dire la manière de traiter les flux audio.

Le tableau 12.5 décrit quelques applications classiques.

**Tableau 12.5 – Applications les plus courantes**

| *Application* | Description |
|---|---|
| *Answer* | Répond à un appel téléphonique entrant. |
| *Background* | Lit un message audio de manière non bloquante. Autrement dit, une saisie d'une ou plusieurs touches par l'appelant est possible en parallèle. Dans ce cas, la lecture du message audio en cours est interrompue. L'extension correspondant aux touches saisies par l'appelant est automatiquement appelée. Cela permet notamment de présenter une information facultative, comme une publicité ou un menu, que l'appelant peut interrompre dès qu'il a pris connaissance de l'option qui l'intéresse. |
| *Dial* | Met en relation l'appelant et l'utilisateur ou le service spécifié en argument de l'application *Dial*. Cette commande est donc utilisée pour affecter un numéro de téléphone à un utilisateur, à un terminal ou à un service.<br>En argument de l'application, il faut indiquer le nom du compte à contacter, préfixé par le protocole de signalisation que le compte utilise (IAX2, SIP, etc.). Cela permet au serveur Asterisk de déterminer dans quel fichier de configuration se trouvent les propriétés du compte appelé (**iax.conf, sip.conf,** etc.). Nous verrons cela un peu plus loin, dans l'exemple 1.<br>Il est possible d'appeler plusieurs correspondants en même temps en mettant leur extension respective en arguments de l'application *Dial,* séparés par le caractère & (esperluette). Par exemple : *Dial (SIP/1001 & SIP/1005)* fait sonner les postes affectés aux comptes *1001* et *1002* en même temps.<br>Il est aussi possible d'ajouter un argument à l'application mentionnant la durée pendant laquelle la tentative d'appel doit s'effectuer. Par exemple : dial (SIP/identifiant, 30 ) permet de faire sonner le téléphone de l'identifiant pendant une durée de 30 secondes. |

**Tableau 12.5 – Applications les plus courantes**

| *Application* | Description |
|---|---|
| *Echo* | Retourne l'écho de ce que l'appelant prononce. Cette application est utile notamment pour tester que les composants audio de réception et d'émission du son sont fonctionnels. |
| *Goto* | Branchement inconditionnel vers un contexte, une extension et une priorité particuliers. |
| *GotoIf* | Branchement conditionnel vers un contexte, une extension et une priorité particuliers lorsqu'une condition est vérifiée. |
| *GotoIfTime* | Branchement conditionnel vers un contexte, une extension et une priorité particuliers lorsqu'une condition temporelle est vérifiée. |
| *Hangup* | Termine une communication. |
| *Meetme* | Invite un utilisateur à participer à une conférence identifiée en argument de l'application *Meetme*. |
| *Playback* | Lit un message audio de manière bloquante : la lecture du message doit se faire intégralement, et l'appelant ne peut interrompre cette diffusion par une saisie de touche sur le clavier téléphonique. |
| *Queue* | Met en attente une communication. |
| *Read* | Lit une variable. L'appelant est invité à entrer une valeur qui est sauvegardée sous forme de variable par le système. Cela permet notamment de demander un mot de passe à l'utilisateur avant d'accéder à un service spécifique. |
| *Record* | Enregistre une communication dans un fichier son. |
| *ResponseTimeout* | Spécifie en argument un délai au bout duquel l'attente du serveur expire. |
| *SayAlpha* | Annonce vocale de caractères textuels spécifiés en argument de l'application |
| *SayDigits* | Annonce vocale de chiffres spécifiés en argument de l'application |
| *SayNumber* | Annonce vocale de nombres spécifiés en argument de l'application |
| *SayPhonetic* | Annonce vocale d'un message phonétique spécifié en argument de l'application |
| *SayUnixTime* | Annonce vocale de l'heure, selon différents formats |
| *SendText* | Envoie un message textuel à l'utilisateur. |
| *SMS* | Envoi et réception de messages instantanés SMS |
| *System* | Exécute une commande système du système d'exploitation. |
| *Transfer* | Transfère l'appel vers un autre poste ou service. |
| *VoiceMail* | Laisse un message vocal. |
| *VoiceMailMain* | Accède au système de messagerie vocale. |
| *Wait* | Attend pendant un certain délai spécifié en argument. |
| *WaitExten* | Attend une saisie d'une extension par l'utilisateur. |

Ces applications peuvent prendre aucun ou plusieurs arguments donnés à la suite du nom de l'application. Lorsque plus d'un argument doit être fourni à une application, les arguments se succèdent avec un caractère de séparation qui peut être indifféremment une virgule ou un pipe (|). Les deux formes suivantes sont donc équivalentes :

```
Mon_application (argument1 , argument2)
Mon_application (argument1 | argument2)
```

Les extraits de plan de numérotation suivants illustrent quelques exemples simples, en les commentant. On suppose que les règles suivantes s'appliquent dans le contexte courant ; nous verrons plus loin comment basculer sur un contexte particulier.

**Exemple 1**

Voici un exemple important dans lequel nous allons affecter un numéro de téléphone à des utilisateurs (ou à des terminaux).

La partie précédente nous avait permis de déclarer des comptes d'utilisateurs (ou des terminaux). Nous avions ainsi été amenés à définir deux comptes SIP (dont les identifiants étaient *david* et *laurent*) et un compte IAX (dont l'identifiant était *1231*). Voici comment effectuer les affectations des numéros attribués aux utilisateurs avec leur compte respectifs.

```
exten => 5551, 1, Dial (SIP/david)
exten => 5552, 1, Dial (SIP/laurent)
exten => 1234, 1, Dial (IAX2/1234)
```

Si un appelant compose le numéro 5551, il sera mis en relation avec l'utilisateur SIP (dont la définition doit impérativement avoir été faite dans le fichier **sip.conf**), qui a pour identifiant *david*. Il en va de même pour l'identifiant *laurent*, avec le numéro 5552, et pour le numéro 1234, qui est lié au compte 1234, lequel utilise le protocole de signalisation *IAX2* (le compte doit être défini dans le fichier **iax.conf**).

**Exemple 2**

En composant le numéro *54321,* les règles suivantes sont enclenchées.

```
exten => 54321, 1, Answer()
exten => 54321, 2, Echo()
exten => 54321, 3, Playback(vm-goodbye)
exten => 54321, 4, Hangup()
```

Lorsque l'appelant compose l'extension *54321,* les étapes suivantes sont lancées :

1. Accepte l'appel et y répond.
2. Renvoie en écho tout ce que dit l'appelant. L'écho se termine lorsque l'appelant saisit la touche dièse.
3. Diffuse un message audio intitulé *vm-goodbye* fourni par défaut pour dire au revoir à l'appelant.
4. Met fin à l'appel.

**Exemple 3**

En appelant un numéro de téléphone prédéfini, nous souhaitons que le serveur Asterisk nous retourne l'heure en cours. Il s'agit en quelque sorte d'une horloge parlante. Les commandes suivantes permettre de fournir ce service :

```
exten => 777, 1, Answer
exten => 777, 2, SayUnixTime(,CET,kM)
exten => 777, 3, Hangup
```

La première ligne accepte l'appel sur le numéro 777, et la dernière le termine. L'application *SayUnixTime* donne l'heure en cours par un message vocal.

## Tester la configuration d'un client

Les équipements des utilisateurs peuvent être divers : téléphone IP, PDA ou ordinateur. Nous allons utiliser dans cette section une solution générique de téléphonie SIP, avec le logiciel grand public gratuit X-Lite.

Simple d'utilisation, ce freeware est disponible en téléchargement (en anglais uniquement pour le moment), sur le site de l'éditeur CounterPath (anciennement XTEN), à l'adresse *http://www.counterpath.com/index.php?menu=download.*

La figure 12.3 illustre l'interface du logiciel. Parmi les autres clients SIP performants, signalons les logiciels SJPhone et LinPhone.

**Figure 12.3**
*Le logiciel client X-Lite*


Pour configurer le client X-Lite, un simple clic droit n'importe où dans l'interface ouvre un menu contextuel permettant d'accéder au menu « SIP Account Settings ».

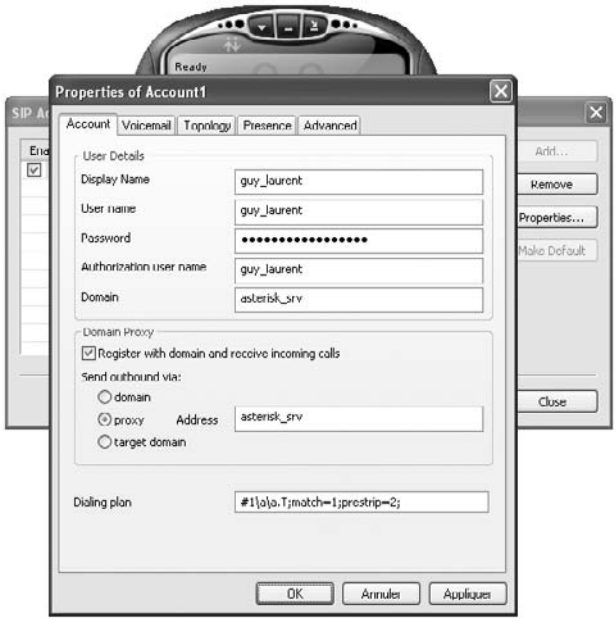
Dans la fenêtre qui s'ouvre, il suffit de renseigner les champs illustrés à la figure 12.4 :

- identifiant affiché pour l'utilisateur (Display Name), pouvant être formé de caractères alphanumériques ;
- identifiant servant à loguer l'utilisateur (User Name) ;
- mot de passe associé (Password) ;
- nom sous lequel l'autorisation d'accès est possible (Authorization user name) ;
- nom de domaine (Domain) ;
- adresse du serveur proxy (Proxy Address), c'est-à-dire le serveur Asterisk lui-même (dans notre cas asterisk_srv).

Il est possible d'indiquer son nom si celui-ci est résolu par un serveur DNS en local ou, à défaut, d'utiliser l'adresse IP du serveur Asterisk.

Pour que l'authentification soit possible, ces valeurs doivent être conformes à celles saisies dans le fichier **sip.conf** du serveur Asterisk.

**Figure 12.4**
*Configuration de X-Lite*




Une fois la configuration achevée, il faut valider ces paramètres en cliquant sur le bouton OK, non sans avoir pris soin de vérifier que la case « Enabled » est cochée pour le compte SIP que l'on vient de créer. Le bouton Close permet de fermer l'interface de gestion des comptes SIP. Le logiciel s'authentifie alors automatiquement auprès du serveur Asterisk mentionné.

Si cette étape s'effectue avec succès, un message « ready » s'affiche, indiquant que les communications sont désormais possibles. À défaut, un message d'erreur explique le motif qui a fait échouer le processus. Dans ce cas, il convient de vérifier la validité des paramètres du compte SIP (nom et mot de passe conformes au fichier **sip.conf**) et l'exactitude de l'adresse IP du serveur Asterisk. Vérifier au besoin qu'un pare-feu ne bloque pas les communications SIP entre le client et le serveur.

## Optimiser les traitements

La section *[globals]* du fichier **extensions.conf** permet de définir des variables, comme dans un langage de programmation.

La syntaxe pour définir une variable est la suivante :

```
Nom_de_variable => Valeur_de_variable
```

L'exemple suivant définit les variables *Numero_de_Guy* et *MusiqueAttente* respectivement initialisées par les valeurs *4321* et */fichier_sons/son_opera.gsm :*

```
[globals]
Numero_de_Guy => 4321
MusiqueAttente  => /fichier_sons/son_opera.gsm
```

Comme nous le constatons, les variables peuvent contenir des valeurs de nature différente, numériques ou textuelles, ou encore des chemins d'arborescence.

Pour utiliser ces variables, il suffit de reprendre leur nom, encadré d'accolades et préfixé par le caractère *$* (dollar). En indiquant *${Numero_de_Guy}* dans le plan de numérotation, c'est la valeur *4321* qui est placée pour cette variable. Si, ultérieurement, le numéro du poste de l'utilisateur *Guy* change, il suffit de remplacer l'unique ligne initialisant la valeur de la variable *Numero_de_Guy* à la nouvelle valeur. Elle est aussitôt prise en compte dans le plan de numérotation sans autre changement.

Il existe des variables spéciales, qui sont préconfigurées par le serveur Asterisk. Il est indispensable de respecter les majuscules et les minuscules dans l'écriture de ces variables. Par exemple :

- EXTEN représente l'identifiant d'extension courante.
- CALLERID(all) représente le nom et le numéro de l'appelant.
- CALLERID(name) représente seulement le nom de l'appelant.
- CALLERID(num) représente seulement le numéro de l'appelant.

- DIALEDTIME représente la durée de l'appel courant.
- DATETIME représente la date courante (son usage est déprécié).
- DIALSTATUS représente l'état de l'appel en cours.

## La directive d'inclusion

Si un fichier est trop volumineux et perd en lisibilité, il est possible de le scinder en plusieurs fichiers. Il suffit pour cela d'utiliser la directive *include,* comme dans les exemples suivants.

Cas 1 : sans inclusion de fichier (un seul fichier) :

```
fichier_1 :                    ligne_1
                               ligne_2
                               ligne_3
                               ligne_4
                               ligne_5
```

Cas 2 : avec inclusion de fichier (deux fichiers) :

```
fichier_1 :          #include fichier_2
                               ligne_4
                               ligne_5

fichier_2 :                    ligne_1
                               ligne_2
                               ligne_3
```

Dans ces exemples, les deux cas sont parfaitement identiques, si ce n'est que le code a été segmenté en deux parties dans le second. La directive optimise la qualité du code en évitant la redondance de code ou en allégeant les fichiers longs.

## Logique de programmation

Parmi les applications disponibles dans la définition du plan de numérotation, on retrouve les classiques procédures de programmation de branchements (ou sauts) conditionnels et inconditionnels. Les applications *Goto()* et *GotoIf()* implémentent ainsi respectivement un branchement inconditionnel et un saut conditionnel.

L'application *Goto* prend comme argument un label spécifiant où doit s'effectuer le branchement. Un label peut être défini exhaustivement par la fourniture de trois éléments : un contexte, une extension et une priorité. Il n'est toutefois pas indispensable de spécifier le

contexte et l'extension. Par défaut, si aucun contexte n'est donné, c'est le contexte courant qui est considéré. De même, si aucune extension n'est fournie, c'est l'extension par défaut qui est prise en compte. La mention de la priorité est le seul élément obligatoire.

Voici un exemple d'application *Goto :*

```
[mon_annonce]
exten => s, 1, Playback(msg_indication)
exten => s, 2, ResponseTimeout (5)
exten => s, 3, WaitExten

exten => #, 1, Goto(mon_service, s, 1)
exten => *, 1, Goto(mon_menu_principal, s, 1)
exten => i, 1, Goto(s, 1)
exten => t, 1, Playback(msg_au_revoir)
exten => t, 2, Hangup
…

[mon_service]
exten => s, 1, …
…
[mon_menu_principal]
…
```

Nous considérons que la communication est gérée par le contexte *[mon_annonce]*. Le serveur réagit en fonction de la saisie de l'utilisateur.

La première extension, *s* (pour *start*), est toujours activée par défaut. La première étape consiste en la diffusion d'un message audio nommé *msg_indication* (qui n'existe pas par défaut), indiquant à l'appelant que ce dernier doit saisir une touche au clavier. La ligne suivante limite l'attente de cette saisie à 5 secondes. La troisième étape de l'extension *s* enclenche l'attente de la saisie.

Les quatre cas suivants peuvent se produire :

- Si l'utilisateur saisit la touche dièse, il est orienté (par l'application *Goto*) vers le contexte *[mon_service]* (premier argument de l'application *Goto*) à l'extension *s* (deuxième argument de l'application *Goto*) et sur l'étape numéro 1 (troisième argument de l'application *Goto*). La ligne sur laquelle le branchement doit s'effectuer est partiellement mentionnée plus bas, sans être implémentée : elle succède à la ligne *[mon_service]*. C'est là que figure la prochaine règle examinée pour le traitement correspondant à la saisie de la touche dièse.

- Si l'utilisateur saisit la touche étoile, le traitement se poursuit sur la première étape de l'extension *s* du contexte *[mon_menu_principal]* (également mentionné plus bas).
- Si l'utilisateur saisit une autre touche que dièse ou étoile, la procédure reprend depuis l'étape précédente, c'est-à-dire à la première étape de l'extension *s* du contexte *[mon_annonce]* (correspondant au contexte courant, qu'il n'est donc pas nécessaire de préciser). L'annonce l'invitant à saisir une touche est à nouveau diffusée, et le système se place en attente de cette saisie.
- Si le délai imparti est écoulé, un message *msg_au_revoir* (n'existant pas par défaut) est diffusé à l'appelant pour lui signifier qu'aucun signal n'a été reçu. La communication s'arrête.

L'application *GotoIf* permet d'effectuer un branchement conditionnel. En argument de la fonction, il suffit de mentionner la condition à réaliser, suivie d'un point d'interrogation puis du label désignant le branchement effectué si la condition est vérifiée, et de terminer par le caractère deux-points suivi du label désignant le branchement qui sera effectué si la condition n'est pas vérifiée.

Il est possible d'omettre l'un des deux labels et de désigner simplement le branchement dans le cas où la condition est (ou n'est pas) réalisée. Il n'est pas possible d'omettre les deux labels simultanément.

Voici un exemple d'application *GotoIf :*

```
exten => 7979, 1,Read (var_secret, rc-code, 4)
exten => 7979, 2, GotoIf ($[${var_secret} = 1234] ? code_valide,s,1 : code_invalide
s, 1)


[code_valide]
…
[code_invalide]
…
```

Ce code spécifie pour l'extension *7979* la lecture d'une variable *var_secret,* qui est sollicitée par un message vocal nommé *rc-code* de 4 chiffres (l'absence de la valeur 4 en argument entraînerait l'acceptation d'autant de chiffres que l'utilisateur le souhaite, la séquence étant terminée et validée par l'appui sur la touche dièse).

Dans la deuxième étape de cette extension, la valeur de la variable saisie par l'utilisateur est comparée à la valeur *1234.* Si l'utilisateur a effectivement saisi cette valeur, la prochaine étape examinée est l'étape 1 de l'extension *s* du contexte *[code_valide].* Si ce n'est pas le cas, on passe à la priorité numérotée 1 de l'extension *s* du contexte *[code_invalide].*

## Optimisation du routage avec les contextes

Les contextes permettent de définir des cadres pour des procédures plus évoluées, ciblées, et contrôlées. Lorsque le code d'un plan de numérotation est segmenté en contexte, il gagne en lisibilité, ce qui simplifie son suivi et sa mise à jour, au contraire d'un code compactant toutes les possibilités avant un seul contexte.

Les contextes sont au plan de numérotation ce que les fonctions sont à la programmation. Ils permettent de factoriser les actions à entreprendre, facilitant d'autant la réutilisation du code. Surtout, le code devenant modulaire, les traitements afférents à une extension particulière, à des catégories d'utilisateurs, à des services particuliers ou à des conditions spécifiques peuvent être regroupés dans une même portion de code, qu'il est plus simple de maintenir ultérieurement.

L'exemple de code suivant :

```
[internal]
exten => 801, 1, Answer()
exten => 801, 2, Echo()
exten => 801, 3, Playback(vm-goodbye)
exten => 801, 4, Hangup()
```

peut être remplacé par le suivant :

```
[internal]
exten => 801, 1, Goto (mon_echo, s, 1)

[mon_echo]
exten => s, 1, Answer ()
exten => s, n, Echo ()
exten => s, n, Playback (vm-goodbye)
exten => s. n, Hangup ()
```

Le second est certes plus long, mais il gagne en clarté, en particulier dans un plan de numérotation de plusieurs centaines de lignes. Il est en outre réutilisable pour d'autres appels.

## Et la vidéo ?

Avec Asterisk, les sessions peuvent gérer des communications à la fois audio et vidéo.

Pour cela, et en supposant que les intervenants effectuent une communication en utilisant le protocole de signalisation SIP, il n'y a qu'une manipulation mineure à faire dans la configuration d'Asterisk.

La modification est à réaliser dans le fichier de configuration **sip.conf** (qui se trouve, par défaut, dans le répertoire **/etc/asterisk**), en décommentant la ligne suivante (c'est-à-dire en retirant les points-virgules qui précèdent cette ligne) :

```
videosupport=yes
```

Dès lors, les intervenants peuvent utiliser le logiciel SIP de leur choix et initier des communications qui couplent la vidéo à la voix.

## Ajouter des sons

Lors de la mise en place d'un serveur vocal, il peut être nécessaire de diffuser à l'appelant un message audio, par exemple, pour indiquer à un utilisateur le nombre de messages déposés sur son répondeur, lui présenter une liste de choix, lui demander de saisir un code d'authentification, ou lui transmettre une annonce informative.

Asterisk offre divers fichiers audio en anglais, espagnol et français synthétisant différents messages génériques. Il est possible d'étendre ces fichiers en ajoutant des sons de son choix.

### Synthèse vocale

Le paquetage **asterisk-core-sounds-fr** (téléchargeable à l'adresse *ftp://ftp.digium.com/pub/telephony/sounds/*) propose plusieurs centaines de fichiers audio en français utilisables avec Asterisk.

Ces messages génériques de quelques secondes sont de bonne qualité. Différents formats de codage sont disponibles (G.711, G.722, G.729, gsm, wav). Le nom complet du fichier à télécharger comporte cette information à la suite du nom du paquetage. Par exemple, en téléchargeant le fichier **asterisk-core-sounds-fr-gsm-1.4.3.tar,** nous obtenons un paquetage au format GSM, dans sa version 1.4.3. Une fois décompressés à l'aide de la commande *tar,* les fichiers doivent être placés dans **/var/lib/asterisk/sounds.**

Pour jouer ces fichiers audio à un appelant, il suffit de lancer la commande *Playback* en mentionnant en argument le nom du fichier à jouer, sans l'extension.

En voici un exemple :

```
exten => 123, 1, Answer()
exten => 123, 2, Playback (hello-word)
exten => 123, 3, Hangup()
```

Si l'appelant compose le numéro d'appel *123,* le fichier audio **hello-word.gsm** est lu et le message audio « salut tout le monde » est diffusé. La communication est ensuite immédiatement coupée.

Un autre cas de figure concerne les nombres. Si le serveur vocal doit renvoyer le nombre de messages reçus ou un temps d'attente ou encore un numéro de téléphone, il n'est pas envisageable de faire figurer tous les nombres préenregistrés possibles. Une synthèse vocale est disponible par le biais des fonctions *SayDigits()* et *SayNumber()*. La première permet de prononcer et distinguer chaque chiffre entier composant un nombre. La seconde prononce le nombre directement (ce nombre doit être compris entre 0 et 99 999 999). Par exemple, la commande *SayDigits(123)* synthétise le message audio « un deux trois », tandis que la commande *SayNumber(123)* synthétise le message audio « cent vingt-trois ».

### Enregistrer ses propres messages

Asterisk offre la possibilité d'enrichir sa base de fichiers audio en l'utilisant comme un magnétophone. Cette fonction est très pratique pour réaliser des annonces spécifiques. Il suffit pour cela d'utiliser la commande *Record* prenant en paramètres un chemin (emplacement du fichier dans le système) associé à un format de fichier.

La syntaxe de cette commande est la suivante :

```
Record (chemin_d'enregistrement:format_de_sauvegarde|temps_son_blanc)
```

Le paramètre *chemin_d'enregistrement* désigne l'emplacement dans l'arborescence du système où le son doit être enregistré. Pour être exploitable par la suite par son nom et pas par son emplacement complet, le fichier doit être placé dans le répertoire **sounds** d'Asterisk.

Le paramètre *format_de_sauvegarde* précise le codec utilisé. Le paramètre *temps_son_blanc* permet d'ajouter un délai additionnel de bruit blanc à la suite du message. L'ajout d'un bruit blanc à un message audio permet à l'appelant d'effectuer ses choix et saisies. Par exemple, si nous souhaitons inviter l'utilisateur à saisir une succession de touches au clavier, il faut lui laisser un temps ni trop court (pour qu'il ait le temps de saisir toutes les touches nécessaires), ni trop long (pour ne pas l'impatienter et lui redonner des directives s'il n'est pas suffisamment réactif).

Ce temps prédéfini est donné en secondes.

En voici un exemple :

```
exten => 456, 1, Answer ()
exten => 456, 2, Playback (conf-unmuted)
exten => 456, 3, Record (/tmp/rec:gsm|7)
exten => 456, 4, Playback (/tmp/rec)
exten => 456, 5, Playback (echo-done)
exten => 456, 6, Hangup ()
```

Lorsque l'utilisateur compose le numéro *456,* un message audio lui joue le message « Vous pouvez parler maintenant ». L'enregistrement du son débute alors dans le fichier **/tmp/rec,** en utilisant le codec GSM. Sept secondes de son blanc y sont ajoutées. Ensuite, le message audio enregistré est rejoué à l'appelant en écho, suivi d'un message *echo-done,* qui indique que l'appel d'écho est terminé, avant de raccrocher.

Cet exemple constitue un bon test pour valider à n'importe quel moment le fonctionnement de son équipement audio. C'est du reste une fonction disponible dans des logiciels tels que Skype (en appelant l'identifiant *echo123*) ou Wengo (numéro d'appel *333*).

#### Utiliser d'autres sons

Il est possible d'utiliser n'importe quel son dans Asterisk, à la condition qu'il respecte les formats de codage supportés. Si ce n'est pas le cas, plusieurs programmes permettent de convertir différents formats non pris en charge par Asterisk en des formats supportés.

C'est le cas notamment du programme SOX, en licence GPL, téléchargeable à l'adresse *http://sox.sourceforge.net.* Après l'avoir installé, il suffit, par exemple, d'effectuer la conversion d'un fichier au format MP3, nommé **test.mp3,** en un fichier au format GSM nommé **test.gsm,** avec la commande suivante :

```
sox test.mp3 -r 8000 -c1 test.gsm resample -ql
```

### Problèmes éventuels avec les modules

Situés dans le dossier **/usr/lib/asterisk/modules,** les modules peuvent être paramétrés au moyen du fichier **/etc/asterisk/modules.conf.**

Par défaut, le paramètre *autoload=yes,* défini dans le fichier **modules.conf,** implique le chargement de tous les modules figurant dans le dossier **modules** d'Asterisk. Certains modules pouvant poser problème, il est possible de changer ce mode et de désactiver les modules au démarrage, en spécifiant la valeur *no* au paramètre *autoload.* Dans ce cas, seuls les modules explicitement mentionnés par la directive *load* sont chargés.

Inversement, il est possible d'activer tous les modules et d'interdire le lancement d'autres modules explicitement mentionnés. Le plus simple en ce cas est de les désactiver. Pour désactiver des modules en particulier, il suffit de conserver la valeur *yes* au paramètre *autoload* et de saisir à la section *[modules]* du fichier **modules.conf** la directive *noload* suivie du module à désactiver.

En voici un exemple :

```
[modules]
noload => cdr_pgsql.so
noload => codec_lpc10.so
```

## Ajouter de nouveaux services

Notre plate-forme Asterisk étant désormais pleinement opérationnelle, nous pouvons mettre en relation tous les utilisateurs entre eux.

Cependant, aucun service évolué n'ayant encore été configuré, nous allons voir comment installer sur le serveur quelques services classiques.

### *Standard vocal automatique (IVR)*

Il est possible d'installer un IVR (Integrated Voice Responder), ou standard vocal automatique, permettant à l'appelant de sélectionner lui-même la personne ou le service avec lequel il souhaite entrer en communication.

Le standard automatique propose des menus à choix multiples conduisant à diverses actions spécifiques, telles que des annonces informatives, le relais vers un service approprié, l'accès à des services de type répondeur ou la mise en relation avec un poste téléphonique particulier.

L'exemple suivant de standard vocal automatique est un cas d'école, puisque incomplet. Si l'utilisateur compose la touche étoile, il est invité à saisir le numéro de poste téléphonique qu'il souhaite contacter. Ce numéro lui est répété avant la mise en communication. Les erreurs et attentes sont également traitées.

```
exten => *, 1, Goto (menu_choix, s, 1)

[menu_choix]
exten => s, 1, Background (enter-ext-of-person)

exten => 1, 1, Playback(you-entered)
exten => 1, 2, Playback(digits/1)
exten => 1, 3, Dial(SIP/guy)
exten => 1, 4, Hangup()

exten => 2, 1, Playback(you-entered)
exten => 2, 2, Playback(digits/2)
exten => 2, 3, Dial(SIP/laurent)
exten => 2, 4, Hangup()

exten => i, 1, Playback (pbx-invalid)
exten => i, 2, Goto (menu_choix, s, 1)

exten => t, 1, Hangup ()
```

La première ligne permet d'accéder au contexte *[menu_choix]* lors de la saisie de la touche étoile. Un message est alors diffusé pour demander à l'appelant de choisir le numéro de poste téléphonique qu'il souhaite joindre. Nous traitons dans cet exemple deux postes, numérotés 1 et 2.

Les possibilités sont les suivantes :

- L'appelant saisit la touche 1 (extension *1*) : un message audio lui répète la touche qu'il vient de saisir, puis le système lance l'appel vers le poste numéro 1, qui est attribué à l'utilisateur *guy* (en signalisation SIP).
- L'appelant saisit la touche 2 (extension *2*) : comme précédemment, après la confirmation de la saisie, la communication vers le poste 2 attribué à l'utilisateur *laurent* est initiée (en signalisation SIP).
- L'appelant saisit une autre touche que 1 ou 2 (extension *i*) : un message audio l'informe de l'invalidité de la saisie, puis l'étape précédente est immédiatement réactivée, l'invitant à saisir une nouvelle touche.
- L'appelant ne saisit aucune touche pendant un délai déterminé (extension *t*) : l'appel se termine aussitôt.

## Conférence

Deux étapes suffisent pour mettre en place une conférence avec Asterisk : créer les salons de conférence virtuelle et y inviter des participants.

### Créer des salons virtuels de conférences (fichier *meetme.conf*)

Pour créer des salons de conférences, il suffit de configurer le fichier **meetme.conf** en ajoutant à la section *[rooms]* le code suivant :

```
conf => numero_de_conference
                                  [ , code_accès_simple ]
                                     [ , code_accès_administrateur ]
```

Le mot-clé *Conf* correspond à une nouvelle salle de conférence virtuelle, définie au minimum par l'indication d'un numéro de salle *(numero_de_conference)*. Il peut être complété optionnellement par un code d'accès que l'utilisateur devra fournir pour accéder à la salle virtuelle et éventuellement d'un code d'accès permettant de reconnaître l'administrateur, auquel des droits de gestion du salon virtuel sont attribués.

Par exemple :

```
conf => 770
```

permet de créer un salon ayant pour identifiant le numéro *770*. Nous pouvons le compléter en remplaçant la ligne précédente par :

```
conf => 770, 12345, 150379
```

Cela crée un salon d'identifiant *770,* auquel les utilisateurs peuvent accéder en indiquant le code *12345* et dont l'administrateur s'identifie par le code *150379.*

### Inviter des participants à la conférence (application *Meetme*)

Pour inviter des participants à entrer dans la salle de conférence, il faut les aiguiller en utilisant le plan de numérotation et l'application *Meetme.*

Pour rediriger une communication vers la conférence précédente, Il suffit d'utiliser l'appel *Meetme (770)* dans le fichier **extensions.conf.**

Par exemple, si l'appelant compose le numéro *770,* l'extension suivante l'invite à rejoindre la conférence *770 :*

```
exten => 770, 1, Meetme (770)
```

Il est possible d'ajouter en second argument de l'application *Meetme* une ou plusieurs des options récapitulées au tableau 12.7 (s'il y en a plusieurs, les options sont indiquées en se succédant sans caractère de séparation).

**Tableau 12.7 – Options de l'application *Meetme***

| Option | Description |
|---|---|
| m | Active le mode *monitor* : les participants peuvent écouter, mais pas parler. |
| p | Un participant peut quitter la conférence en pressant la touche dièse. |
| t | Active le mode *talk* : les participants peuvent parler, mais pas écouter. |
| v | Active le mode *video.* |
| q | Mode silencieux *(quiet)* . aucun son n'est émis lorsque des utilisateurs entrent dans la conférence ou en sortent. |
| d | Ajoute une conférence dynamiquement. |
| M | Active une musique d'attente lorsqu'il n'y a qu'un seul participant à la conférence. |
| b | Lance le script AGI spécifié dans la variable *MEETME_AGI_BACKGROUND* (celle-ci doit avoir été initialisée auparavant). |

## Le service de messagerie audio (fichier voicemail.conf)

Le service de messagerie audio se met en place très simplement *via* la configuration de seulement trois fichiers : le premier permet de définir les paramètres du compte de messagerie vocale, le deuxième d'accéder à la boîte vocale créée, et le troisième de signaler à un utilisateur tout nouveau message vocal reçu.

Le principe du service de messagerie audio consiste à définir un numéro de boîte vocale associé à un utilisateur. Cela permet notamment à un même utilisateur d'avoir plusieurs numéros de téléphone (professionnel, personnel, autre) et une seule messagerie unifiée, archivant les messages qui lui sont destinés, que ces derniers soient personnels, professionnels ou autres.

Dans un cadre professionnel, il est aussi possible de permettre à plusieurs utilisateurs d'avoir un numéro de téléphone spécifique, tout en leur assignant une messagerie unifiée. Un message laissé sur le répondeur unique peut de la sorte être traité par la première personne disponible dans le service.

Les sections qui suivent montrent comment créer des boîtes vocales, les affecter à des utilisateurs et signaler la présence de nouveaux messages audio.

### Créer des comptes de messagerie audio (fichier *VoiceMail.conf*)

La première étape consiste à créer les boîtes vocales. Le fichier **VoiceMail.conf** comporte à cet effet deux sections distinctes : *[general],* qui permet de paramétrer les options des comptes, et *[default],* qui décrit les comptes à créer.

En voici un exemple :

```
[general]
format=gsm
attach=yes
fromstring=Ma Messagerie Vocale

[default]
3535 => 15155, guy_laurent, guy_laurent@fai_home.fr
```

La section *[général]* permet de définir différents paramètres de configuration, notamment les suivants :

- *format :* définit le format d'encodage des messages audio enregistrés (par défaut *wav49, gsm* et *wav*).
- *maxmessage :* spécifie la durée maximale (en secondes) d'un message. La valeur par défaut est *0,* signifiant qu'il n'y a pas de durée maximale.
- *minmessage :* spécifie la durée minimale (en secondes) d'un message. La valeur par défaut est *0,* signifiant qu'il n'y a pas de durée minimale.

- *maxmsg :* spécifie le nombre maximal de messages qui peuvent être laissés sur la boîte vocale. La valeur par défaut est *100.*
- *review :* autorise l'appelant à écouter et modifier le message laissé. La valeur par défaut est *no.*
- *attach :* indique si les messages vocaux laissés sur une messagerie sont également envoyés par e-mail aux utilisateurs concernés. La valeur par défaut est *no.*
- *maxlogins :* indique le nombre maximal de tentatives d'accès infructueuses autorisé.
- *emailsubject :* mentionne le sujet de l'e-mail notifiant le message vocal.
- *mailcmd :* indique chemin de la commande utilisée pour l'envoi des e-mails. La valeur par défaut est *usr/sbin/sendmail –t.*
- *fromstring :* spécifie un nom avec lequel l'e-mail notifiant le message vocal est envoyé. La valeur par défaut est *Asterisk PBX.*
- *emailbody :* indique le contenu de l'e-mail signalant la réception de nouveaux messages vocaux. Il est possible d'utiliser des variables dans la définition de ce paramètre.

  Dans l'exemple suivant, le nom de l'utilisateur et le nombre de messages sont automatiquement remplacés par les variables correspondantes :

```
emailbody=Bonjour ${VM_NAME}, vous avez ${VM_MSGNUM} message(s) dans votre boîte
vocale.
```

Dans notre exemple, après la section *[general],* la section *[default]* spécifie les comptes à créer et leurs propriétés. Nous avons créé une seule boîte vocale identifiée par le numéro *3535.* Son mot de passe, *15155,* permet à son possesseur d'accéder à sa messagerie. Cette dernière est affectée à l'utilisateur *guy_laurent* ayant pour adresse de messagerie *guy_laurent@mon_fai.fr* (c'est à cette adresse que seront envoyés les messages vocaux laissés sur la messagerie de l'utilisateur si le paramètre *attach* défini précédemment est activé à la valeur *yes*).

### Laisser un message et écouter ses messages (applications *VoiceMail* et *VoiceMailMain*)

Nous considérons ici deux cas de figure, selon qu'un utilisateur souhaite laisser un message sur la boîte vocale d'un autre utilisateur ou consulter ses propres messages sur sa propre boîte vocale.

#### Accéder à la boîte vocale d'un correspondant

Dans le premier cas, sans réponse de l'appelé au bout d'un certain temps, il faut enclencher une procédure effectuant le basculement de l'appel vers la messagerie du correspondant.

La fonction à utiliser pour cela est *VoiceMail,* qui prend comme argument le numéro de la boîte vocale, ainsi que, optionnellement, le contexte associé. Cet argument est donné sous la forme *numéro_de_boite_vocale@contexte_associé,* par exemple :

```
Voicemail(1000@default)
```

Dans l'exemple suivant inséré dans le plan de numérotation (fichier **extensions.conf**), l'appel est initié vers l'utilisateur *guy_laurent* en composant le numéro *1235*. Au bout de 50 secondes sans réponse, la messagerie affectée à *guy_laurent* et identifiée par le numéro de boîte vocale *3535* est enclenchée :

```
exten => 1235, 1, Dial (SIP/guy_laurent, 50, tm)
exten => 1235, 2, Voicemail (3535@default)
exten => 1235, 3, Hangup
```

La fonction *VoiceMail* prend en charge le service de répondeur en invitant l'appelant à parler et en enregistrant le message. Ce dernier est accessible par l'appelé en consultant sa boîte vocale.

### Consulter sa boîte vocale

Pour gérer la consultation des messages, il suffit de mettre en place un numéro d'appel spécialement dédié à la consultation de la messagerie. La fonction *VoiceMailMain* permet de lancer la messagerie souhaitée. Comme la fonction *VoiceMail,* elle prend pour argument le numéro de boîte vocale et optionnellement le contexte associé.

Dans l'exemple suivant, en composant l'extension *800,* l'appel est directement redirigé vers la consultation de la messagerie correspondant à la boîte vocale identifiée par le numéro *3535* (correspondant au compte de messagerie vocale de l'utilisateur *guy_laurent*) :

```
exten => 800, 1, Answer
exten => 800, 2, VoiceMailMain(3535@default)
exten => 800, 3, Hangup
```

La fonction *VoiceMailMain* prend en charge le service de consultation en demandant notamment le mot de passe de l'utilisateur, puis en lançant la lecture, la suppression et plus généralement la gestion des messages audio.

Cette implémentation peut être optimisée. Par exemple, au lieu d'affecter le numéro d'appel *800* exclusivement à la boîte vocale de l'utilisateur *guy_laurent,* il serait préférable, comme cela se fait couramment en téléphonie classique, de disposer d'un numéro d'appel unique pour gérer la messagerie de tous les abonnés et d'utiliser une identification et authentification pour accéder à la boîte vocale désirée.

#### Signaler la présence d'un nouveau message audio

Pour signaler aux utilisateurs la présence d'un nouveau message audio dans leur boîte vocale, il suffit d'associer cette dernière au compte d'un utilisateur. Par exemple, si un utilisateur utilise le protocole de signalisation SIP, son compte est créé dans le fichier **sip.conf,** et il suffit d'ajouter à la section *[guy_laurent]* de ce compte la ligne suivante :

```
mailbox=3535@default
```

## Aller plus loin avec Asterisk

Les possibilités d'exploitation du serveur Asterisk sont telles que le logiciel n'a pas à rougir de la comparaison avec ses équivalents PBX traditionnels, souvent hors de prix et de portée des particuliers.

Dans ce chapitre, nous n'avons évoqué qu'un fonctionnement standard, illustrant les usages les plus courants, mais quantité d'autres possibilités sont envisageables. On pourrait presque dire que tout ce qu'un PBX physique traditionnel sait faire, Asterisk peut le faire aussi.

Nous mentionnons dans cette section quelques autres fonctionnalités parmi les plus remarquables offertes par Asterisk.

### *Connecter Asterisk à un fournisseur SIP*

Il existe de nombreux acteurs qui proposent à leurs clients de communiquer en utilisant le protocole de signalisation SIP. C'est le cas, par exemple des fournisseurs d'accès français Free et Neuf Télécom, mais aussi de la société Wengo avec son logiciel WengoPhone dont nous avons déjà parlé dans un précédent chapitre, ou encore de VoIPDiscount, également évoqué précédemment. Bien souvent, ces comptes sont associés à des conditions tarifaires très avantageuses, notamment la gratuité des appels dans plusieurs dizaines de pays. Pourquoi ne pas faire profiter Asterisk de ce compte ?

L'idée ici serait alors de disposer d'un compte SIP que nous procure l'une de ses sociétés, et de configurer Asterisk avec celui-ci. De cette manière tous les téléphones reliés au serveur Asterisk pourront bénéficier des mêmes conditions tarifaire de leur compte SIP. Les avantages sont multiples :

- Tous les téléphones connectés à Asterisk peuvent tirer profit du compte SIP, même s'ils ne sont pas compatibles SIP, puisque Asterisk sert de passerelle.

- Les services activés sur Asterisk restent disponibles dans le cadre des communications effectuées *via* le compte SIP (journalisation des appels, enregistrement du carnet d'adresses, etc.).
- Les utilisateurs connectés n'ont pas à configurer leur logiciel avec le compte SIP (ils n'ont même pas besoin de le connaître).
- Si le fournisseur SIP propose un numéro d'appel entrant, il n'est pas nécessaire d'avoir un logiciel SIP spécifique qui soit actif, ni même un téléphone de VoIP compatible SIP pour recevoir les communications : Asterisk pourra être configuré pour recevoir tous les appels entrants vers ce numéro d'appel et les rediriger vers n'importe quel téléphone.

On considère dans la suite qu'on dispose d'un serveur Asterisk fonctionnel et d'un compte SIP (identifiant, mot de passe, adresse du serveur SIP à contacter et éventuellement numéro de téléphone si le fournisseur SIP le propose). On ne doit s'assurer que l'on dispose des paramètres SIP correctes (souvent, l'utilisateur dispose d'un compte auprès du fournisseur, qui n'a rien à voir avec son compte SIP, mais sert, par exemple, à la facturation de son compte sur Internet ou à d'autres fonctionnalités : c'est notamment le cas avec le logiciel WengoPhone ou les utilisateurs se connectent au logiciel avec des paramètres qui ne sont pas des paramètres SIP).

Nous allons mettre en œuvre cette fonctionnalité, en réalisant plusieurs étapes : d'abord on configurera le serveur Asterisk comme étant un client SIP, puis on déclarera le compte SIP dont on dispose et enfin on redirigera les appels sortants vers le fournisseur SIP, et les appels entrants vers le téléphone de son choix.

Pour réaliser la première étape, on ajoute une ligne dans le fichier sip.conf, à la section [general]. Cette ligne va permettre de s'enregistrer auprès du fournisseur SIP comme étant l'utilisateur, et comme étant actif dans le réseau (donc pouvant appeler et recevoir des communications). Elle doit avoir le format suivant :

```
register => identifiant[:mot_de_passe]@adresse_fournisseur_sip[:port][/param]
```

L'extension à saisir pour faire d'Asterisk un client est *register*. On lui associe trois informations : le nom de l'utilisateur (*identifiant*), son mot de passe presque toujours obligatoire (*mot_de_passe*), et on mentionne le nom (ou l'adresse IP) du serveur SIP que le fournisseur SIP doit fournir (adresse_*fournisseur_sip*). Eventuellement on peut préciser un port spécifique si le fournisseur SIP l'exige (*port*), et un paramètre que le fournisseur SIP pourra interpréter.

Un exemple d'utilisation est donné dans ce qui suit :

```
register => david:d@v1d@sip.voipdiscount.com
```

Dans un second temps, et toujours dans le fichier sip.conf, après la section [global], on déclare une section qui définira les propriétés du compte SIP :

```
[nom_fournisseur_sip]
type=peer
username=identifiant
secret= mot_de_passe
fromuser=identifiant
callerid=identifiant <numero_tel>
fromdomain=adresse_fournisseur_sip
host=adresse_fournisseur_sip
nat=yes
canreinvite=no
context=appels_entrants
```

On retrouve essentiellement des mêmes éléments déjà présentés, et qu'il faut remplacer par les paramètres du compte (ces paramètres sont en italiques), avec, optionnellement, un numéro de téléphone (*numero_tel*) si le fournisseur SIP le propose. Les paramètres *nat* et *canreinvite* sont également à adapter (selon qu'on se trouve effectivement derrière un nat ou non). L'intitulé de la section est quelconque (ici *nom_fournisseur_sip*). Le contexte nommé *appels_entrants* sera défini plus loin.

Puis, dans la section [internal] du fichier *extensions.conf*, on ajoute une ligne pour rediriger les appels sortants (donc que les utilisateurs d'Asterisk émettent) vers le serveur SIP du fournisseur :

```
exten => _0., 1, Dial(SIP/${EXTEN}@adresse_fournisseur_sip)
```

L'identifiant d'extension « *_0.* » permet de filtrer tous les appels dont le numéro commence par le chiffre 0. On remarque l'utilisation de la variable spéciale EXTEN, qui correspond au numéro de téléphone saisi par l'appelant.

Enfin, et optionnellement, si le fournisseur SIP propose à ses clients un numéro de téléphone, alors toujours dans le fichier extensions.conf, on ajoute une section pour rediriger les appels entrants sur ce numéro (qu'on notera comme précédemment numero_tel) vers le compte de son choix (ici le compte *nom_compte*, à remplacer par *nom_compte*) :

```
[appels_entrants]
exten => numero_tel, 1, Answer ()
exten => numero_tel, 2, Dial(SIP/nom_compte)
exten => numero_tel, 3, Hangup ()
```

Dans ce cas, lorsque des utilisateurs cherchent à joindre le numéro *numero_tel*, l'appel est automatiquement rediriger vers le compte *nom_compte* en utilisant le protocole de signalisation SIP (qui doit être défini dans le fichier sip.conf).

Bien sûr, cette possibilité est à restreindre au seul cercle privé. Elle entrerait en violation avec les conditions d'utilisation du contrat si elle était exploitée à des fins mercantiles, ou tout simplement en dehors du cercle privé.

## AGI (Asterisk Gateway Interface)

Asterisk est un logiciel totalement ouvert, à la fois par ses sources, qui sont disponibles en téléchargement et que les programmeurs peuvent enrichir et personnaliser au sein de la communauté, et par le développement de logiciels tiers et l'interaction avec eux.

Les développeurs d'Asterisk facilitent le travail des autres programmeurs en leur proposant une interface générique de contrôle et de gestion du serveur Asterisk, appelée AGI (Asterisk Gateway Interface). N'importe qui peut donc développer une application, dans le langage de programmation de son choix, et la personnaliser à son grès afin d'interagir avec le serveur Asterisk.

Peu de constructeurs offrent ce degré d'ouverture et de compatibilité. Le plus souvent, les utilisateurs doivent se contenter de ce que leur proposent leurs équipementiers, car l'implémentation logicielle, en plus d'être fermée, est protégée dans son code source et n'offre généralement aucun système d'interface comparable à l'AGI.

## Trixbox

Trixbox (anciennement Asterisk@home) est une distribution complète et libre du serveur Asterisk qui à la particularité d'être accessible à partir d'un CD bootable. Elle peut être téléchargée sur la page de l'éditeur, à l'adresse *http://www.trixbox.org.*

Actuellement disponible uniquement en version anglaise, ce logiciel permet de se faire une idée d'Asterisk avant de l'adopter et sans se lancer dans des procédures d'installation et de configuration.

Par défaut, tout est prévu pour fonctionner au lancement. Il est toutefois possible de modifier les configurations de base en suivant les indications fournies dans ce chapitre. Pour cela, il suffit d'arrêter le serveur Asterisk puis de remplacer le contenu des fichiers du répertoire **/etc/asterisk** par ceux indiqués aux sections précédentes de ce chapitre.

Certains modules peuvent être désactivés s'ils sont inutiles ou qu'ils posent problème (option *noload* dans le fichier **modules.conf**).

## Communiquer avec le protocole IAX

Le projet Asterisk a donné naissance à un second projet, appelé IAX (Inter Asterisk eXchange). Celui-ci définit un protocole permettant l'interconnexion entre serveurs Asterisk, mais également la communication entre un client et un serveur Asterisk.

Initialement, le protocole IAX a été développé par le concepteur d'Asterisk, Mark Spencer, de la société Digium. Il est aujourd'hui maintenu par la société Digium et est disponible dans sa deuxième version IAX2, laquelle fait l'objet d'une proposition de normalisation à l'IETF.

Pour être convaincante dans un contexte où la concurrence entre les protocoles H.323 et SIP est déjà importante, la philosophie proposée par IAX diffère sur deux points importants :

- Traversée transparente des passerelles NAT et des pare-feu. Contrairement aux protocoles SIP et H.323, qui n'assurent que la fonction de signalisation et se combinent généralement à RTP pour la fonctionnalité de transport des flux, le protocole IAX est à la fois un protocole de transport et un protocole de signalisation. Cela lui permet plus facilement de traverser les pare-feu et de supporter les translations d'adresses IP (NAT) dans un réseau. Ses flux n'utilisent qu'un port fixe et unique (le port 4569) et peuvent de la sorte être aisément identifiés.
- Utilisation réduite de la bande passante. Si H.323 et SIP sont prévus pour le multimédia en général, IAX a été conçu spécifiquement pour le problème du transport et de la signalisation de la voix, en écartant les considérations plus générales des applications multimédias. Le protocole IAX répond ainsi à des objectifs simples et bien délimités. Bien qu'il n'exclue pas *a priori* le traitement de flux vidéo, il s'intéresse avant tout aux flux audio et optimise les paramétrages des flux en tenant compte des contraintes et des spécificités de ces flux audio.

IAX est un protocole puissant, qui propose des solutions efficaces aux deux problèmes importants rencontrés par H.323 et SIP et permet les communications entre serveurs Asterisk. Il souffre toutefois de l'inconvénient de ne pas être normalisé. De plus, il n'optimise que le traitement des flux téléphoniques, alors que H.323 comme SIP sont plus généralistes et peuvent s'appliquer au transfert de la vidéo.

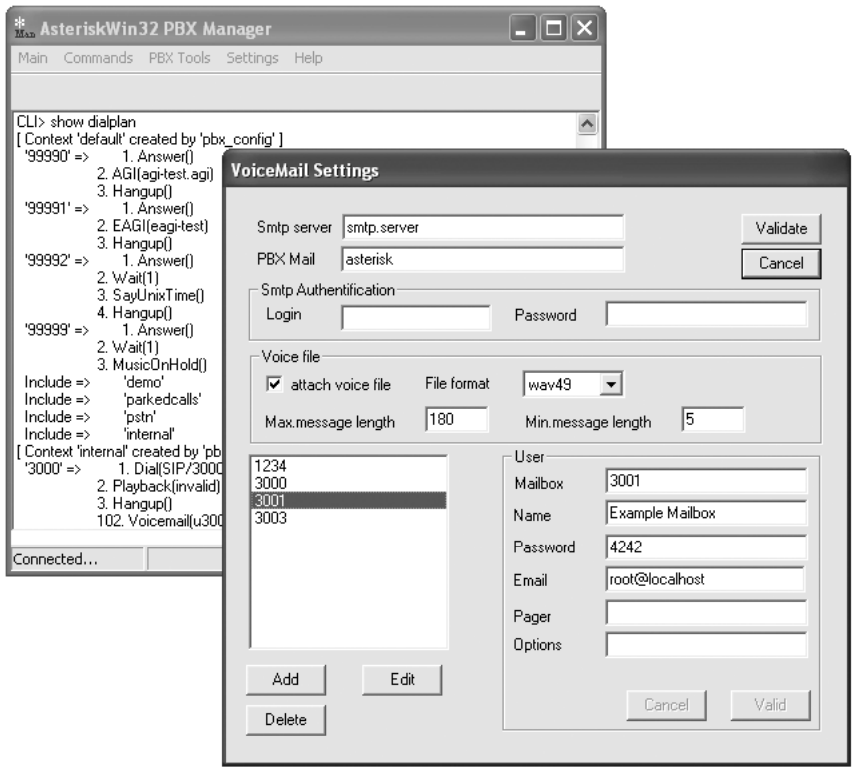
## Asterisk sous Windows

Une version du serveur Asterisk a été développée sous plate-forme Windows. Nommée asteriskwin32, elle est téléchargeable à l'adresse *www.asteriskwin32.com.* Bien qu'elle soit beaucoup moins performante que le logiciel original sous Linux, cette solution constitue un excellent moyen de se familiariser avec les concepts généraux d'Asterisk.

L'utilisateur dispose d'une interface conviviale lui permettant d'accéder à toutes les fonctionnalités du logiciel. On y trouve notamment une préconfiguration initiale du plan de numérotation, qui peut être visualisée en sélectionnant la fonction Dialplan du menu Commands. Comme sous Linux, la modification s'effectue en éditant le fichier **extension.conf** placé dans le répertoire **/etc/** du répertoire spécifié lors de l'installation du logiciel.

De même, des comptes d'utilisateurs sont déjà créés (SIP et IAX uniquement), et les principaux services sont préconfigurés. Pour personnaliser ces services, nul besoin d'aller modifier des fichiers. Il suffit d'utiliser l'interface, comme l'illustre la figure 12.5 représentant la configuration du service de messagerie téléphonique.

**Figure 12.5**
*Configuration du service de messagerie audio sous Windows*




## La concurrence

Si Asterisk est le plus connu des PBX logiciel, il n'est pas le seul. Deux autres logiciels, également libres, Vocal et SIP-X, ont une vocation semblable à celle d'Asterisk.

### Vocal

Développé en 1999 par la société Vovida, Vocal (Vovida Open Communication Application Library) est soutenu par d'importants industriels tels que Cisco, qui en a fait l'acquisition en novembre 2000.

Solution de téléphonie complète et multiplate-forme (BSD, Linux, Windows, Solaris), Vocal inclut des interfaces d'administration du système, un serveur de politiques exploitant le protocole COPS (Common Open Policy Service) et un large choix de fonctionnalités téléphoniques.

La communauté Vocal demeure cependant nettement moins active que celle d'Asterisk. Le produit lui-même est moins abouti en termes d'intégration avec les codecs et protocoles existants.

Le site de l'éditeur *(http://www.vovida.org)* permet d'obtenir davantage d'information sur ce PBX-IP.

### SIP-X

Parrainé par la société Pingtel, SIP-X est le dernier-né des autocommutateurs libres. À la manière de Digium pour Asterisk, Pingtel assure les services de maintenance et de mise à jour du logiciel. Plus généralement, les développeurs sont regroupés au sein du groupe SIPFoundry.

Pour fonctionner, SIP-X requiert l'utilisation de logiciels clients ou de terminaux compatibles SIP exclusivement. Il n'est donc pas aussi large d'utilisation qu'Asterisk.

Plus d'informations peuvent être trouvées sur le site de l'éditeur, à l'adresse *http://www.sipfoundry.org/sipX/index.htm.*

## Conclusion

Ouvert à tous, gratuit, simple à utiliser, puissant et performant, Asterisk a vraiment de quoi séduire, et même rivaliser avec les équipements professionnels. En fait, les vrais concurrents d'Asterisk ne sont pas les autres PBX logiciels, mais les PBX hardware eux-mêmes.

En effet, si les PBX hardware sont chers, ils demeurent performants et fiables. Surtout qu'ils disposent généralement d'un support technique appréciable. Au moindre problème, un technicien peut intervenir dans des délais très courts, ce qui rassure évidemment les entreprises, lesquelles ne sont pas toujours prêtes à faire des économies s'il faut, pour cela, tenter le pari d'une solution ouverte. Ces solutions libres, en effet, peuvent fournir des outils performants et très bien documentés, mais généralement sans procurer la garantie d'un service après-vente, pourtant rassurante lorsqu'on exploite une infrastructure destinée à la téléphonie sur IP.

Il faudra donc certainement encore du temps avant qu'Asterisk gagne le même statut que ces concurrents et dispose d'un capital de confiance aussi fort chez les professionnels. De nombreuses sociétés œuvrent cependant dans ce sens et proposent d'ores et déjà un service de bout en bout assurant la fourniture matérielle et logicielle, ainsi que l'installation et la maintenance du logiciel.

Dans un secteur en pleine mutation, où le monde RTC s'efface au fur et à mesure que le monde IP prend sa place, Asterisk influence d'ores et déjà les stratégies des équipementiers en montrant la voie.

# 13

# La téléphonie chez les fournisseurs d'accès

Le haut débit à l'accès est devenu une nécessité dans un monde où la quantité et la qualité des informations à transporter augmentent sans discontinuer. Un débit de l'ordre du mégabit par seconde semble être une valeur minimale pour réaliser des accès dits à haut débit. Avec l'arrivée de la télévision et de sa version haute définition associée à plusieurs chaînes de télévision simultanées, il faut pouvoir compter aujourd'hui sur une cinquantaine de mégabits par seconde.

Ce chapitre s'intéresse aux accès haut débit terrestres pour les particuliers et les petites et moyennes entreprises, et plus précisément à l'intégration de la parole téléphonique dans ces environnements. Ces accès comprennent quatre types : la ligne téléphonique par le biais d'un modem *x*DSL, le câble CATV associé à un modem câble, la fibre optique et l'accès Wi-Fi en Quadruple-Play.

## Les accès *x*DSL

Les modems *x*DSL permettent d'utiliser les paires métalliques du réseau d'accès pour réaliser une boucle locale à haut débit. Le débit dépend fortement de la qualité du câble utilisé et de la distance à parcourir. Plusieurs catégories de modems *x*DSL sont commercialisées, la lettre *x* permettant de les différencier.

Les modems ADSL (Asymmetric Digital Subscriber Line) sont les plus répandus. Leurs vitesses sont dissymétriques, plus lentes entre le terminal et le réseau que dans l'autre sens. En règle générale, le sens montant est quatre fois moins rapide que le sens descendant. Les vitesses sur le sens descendant peuvent atteindre 2 Mbit/s pour une distance de