

PROJET DE SEMESTRE : SIP BILLING

Foli Kodjo GABA
Juin 2007

En collaboration avec YOURROUTE Sarl
Responsable : Mme Sona Gabrielyan

Laboratoire des Télécommunications
Professeur EIG : Andrés Revuelta

Table des matières

1	Introduction	2
2	Cahier des charges et planning	3
2.1	Cahier des charges	3
2.2	Planning	4
3	La téléphonie sur IP	5
3.1	Présentation	5
3.2	Les principes de base du transport de la voix sur IP	5
3.3	Les Protocoles de communication [3]	5
3.3.1	Le protocole SIP	6
3.3.2	Le protocole IAX	8
3.3.3	Le protocole H.323	8
3.4	La facturation avec le logiciel A2billing	9
3.4.1	L'Asterisk Gateway Interface	9
3.4.2	Les principes de base de A2billing	9
4	L'environnement de test et les tests effectués	11
4.1	L'environnement de test	11
4.2	Résultat des tests	11
4.2.1	Nombre d'utilisateurs dans la base	11
4.2.2	Assignation de prix par destination	11
4.2.3	Limitation de crédit	11
4.2.4	Blocage si crédit nul	13
4.2.5	Interruption d'appel si le crédit est dépassé	13
4.2.6	Low Cost Route (LRC)	13
4.2.7	Détection de la destination par préfixe	13
4.2.8	Statistiques	13
4.2.9	Méthodes de paiement	13
5	Étude économique succincte du projet	14
5.1	Coûts de déploiement	14
5.2	Comparaison avec une solution payante	14
6	Conclusion	16
7	Remerciements	17

A	Mise en place de l'environnement de test	18
A.1	Infrastructure matérielle de test	18
A.2	Infrastructure logicielle de test	18
A.3	Installation du IPBX TrixBox	18
A.4	Configuration du TrixBox [1]	19
A.5	Installation de A2Billing	25
A.6	Configuration de A2Billing	27
A.7	Configuration des Softphones	29
A.8	Detail et facture des appels	30
A.9	Configuration du HardPhone Cisco 7960	30
A.10	Configuration du Routeur Cisco 1760	30
	Bibliographie	33

Table des figures

1.1	Auto commutateur classique et Commutateur IP	2
3.1	Échantillonnage d'un signal continu par un train d'impulsion de Dirac	6
3.2	Le trapèze SIP	7
3.3	Scénario d'un appel simple par SIP [2]	7
3.4	Procédure d'un appel A2Billing	10
4.1	Environnement de test	12
A.1	Page d'accueil TrixBox	19
A.2	Invite d'authentification	20
A.3	Interface FreePBX	20
A.4	Tools de FreePBX	20
A.5	Module Admin de Tools	20
A.6	Activation du compte IAX sur Free World Dialup	21
A.7	Name et Number Trunk	21
A.8	Outgoing Dial Rules	22
A.9	Outgoing Settings	22
A.10	Incomming settings	23
A.11	Registration	23
A.12	Registration	24
A.13	Installation réussie	27
A.14	Rajouter une Ratecard à un Tarifgroupe	28

Chapitre 1

Introduction

L'aspect économique est un facteur déterminant dans le choix des solutions matérielles et logicielles. Sur le marché des commutateurs téléphoniques on a :

- Les systèmes à commutation de circuits classiques et payants, qui ont fait leurs preuves jusqu'ici en offrant une bonne qualité de service ;
- Les systèmes à commutation de paquets, payants ou non, qui promettent une réduction des coûts de déploiement, de maintenance et d'exploitation par rapport aux commutateurs classiques ;
- Les commutateurs hybrides qui intègrent les deux systèmes précités .

Est – il possible de trouver sur le marché une solution entièrement "Open Source" intégrant un commutateur IP et un logiciel de facturation ?

Ce projet né de la collaboration entre "YOUROUTE Sarl" filiale de l'opérateur Suisse offrant les meilleurs prix en téléphonie sur IP "SWITZERNET", et le laboratoire de télécommunications, de l'École d'Ingénieurs de Genève, tente de répondre à cette question.

Ce rapport sera présenté de la manière suivante :

Le chapitre 2 introduira le cahier des charges et le planning d'exécution du projet ;

Le chapitre 3 traitera du contexte général de la téléphonie sur IP, en insistant sur protocoles et la facturation ;

Le chapitre 4 abordera l'environnement et les tests effectués en considérant les spécifications du cahier des charges ;

Enfin le chapitre 5 présentera une étude économique très succincte du projet.

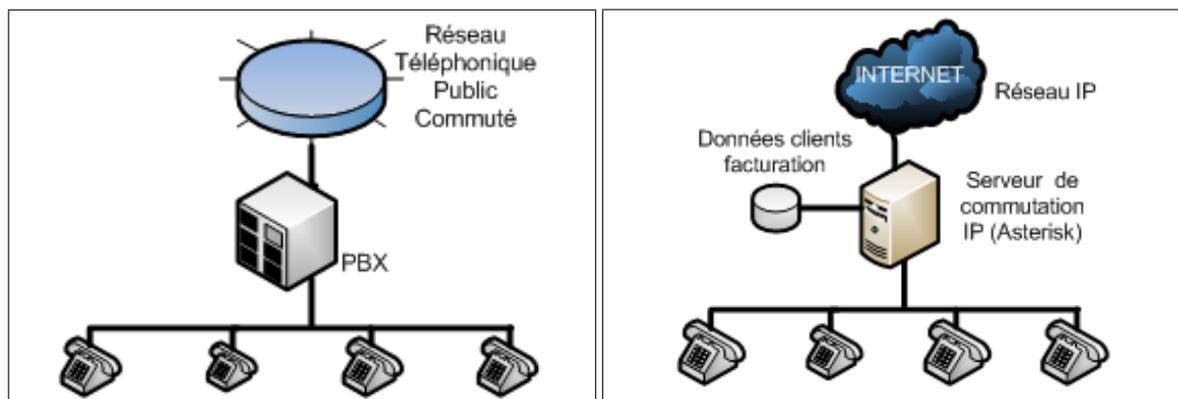


FIG. 1.1 – Auto commutateur classique et Commutateur IP

Chapitre 2

Cahier des charges et planning

Suite à la rencontre avec "Youroute Sarl", le 23 Février 2007, le cahier des charges et le planning suivant ont été établis.

2.1 Cahier des charges

Pour répondre à la question posée dans l'introduction, un choix motivé par la simplicité d'installation a été effectué parmi les solutions "Open source" de commutateurs VoIP (IPBX), et de logiciels de billing (facturation). Le cahier des charges suivant a été élaboré puis approuvé par l'entreprise, ayant proposé le projet.

1. Mise en route du Paquetage TrixBot ;
2. Mise en route et configuration du logiciel de billing A2Billing ;
3. Étude des fonctions suivantes sur TrixBot-A2Billing :
 - Nombre d'utilisateurs dans la base ;
 - Assignation de prix par destination ;
 - Limitation de crédit ;
 - Blocage si crédit nul ;
 - Interruption d'appel si le crédit est dépassé ;
 - Détails des appels ;
 - LCR (Least Cost Route) ;
 - Détection de la destination par préfixe ;
 - Statistiques ;
 - Méthodes de paiement (E-commerce module).
4. Recherche d'autres "Open Source" et Étude des possibilités de développement pour les fonctions non prises en charges par TrixBot - A2Billing dans la liste figurant au point 3.

Date	Travaux Prévus
23.02.2007	Visiter Youroute Sarl - Cahier des charges
02.03.2007	Installation du resau de test
09.03.2007	Installation et configuration d' A2Billing
16.03.2007	Test du Point N° 3 du cahier des charges
23.03.2007	Test du Point N° 3 du cahier des charges
23.03.2007	Test du Point N° 3 du cahier des charges
06.04.2007	Rapport N°1 sur les tests
20.04.2007	Configuration IP phone CISCO 7960
27.04.2007	Étude des possibilités de développement
04.05.2007	Connexion au réseau ISDN
11.05.2007	Connexion au réseau ISDN
18.05.2007	Connexion au réseau ISDN
25.05.2007	Édition du rapport
01.06.2007	Marge de retard
08.06.2007	Rendu du rapport
15.06.2007	Présentation finale

TAB. 2.1 – Planning

2.2 Planning

Le travail devra être effectué tous les vendredis ouvrables, à raison de 8 heures par jour, du 23 février au 8 juin 2007. La présentation finale est prévue pour le 15 Juin 2007 à 13h00. Le planning du tableau 2.1 été retenu après quelques modifications.

Chapitre 3

La téléphonie sur IP

3.1 Présentation

Les ingénieurs travaillant dans le domaine de la voix sur IP plaisaient parfois en disant que s'ils avaient proposé de transporter de la voix sur des réseaux IP, il y a quelques années, ils auraient sans doute été pris pour des fous. Grâce à l'apparition de processeurs de plus en plus performants, ayant de grande capacité d'exécution d'algorithmes, il est devenu possible de faire de la téléphonie sur IP.

3.2 Les principes de base du transport de la voix sur IP

Le principe de base de cette technologie, est la mise en paquets (échantillonnage d'un flux régulier d'information) pour le transport sur le réseau IP. L'échantillonnage repose sur le postulat d'un Ingénieur Électricien Harry Nyquist¹ énoncé dans les années 20 et prouvé en 1949 par Claude Shannon :

En échantillonnant un signal, la fréquence d'échantillonnage doit être au moins deux fois plus grande que la fréquence maximale du signal pour être capable de reconstruire parfaitement le signal d'origine à partir de la version échantillonnée. En terme mathématique on écrira :

$$f_{\text{échantillonnage}} > 2 * \text{Bande - Passante}_{\text{utile}}$$

Pour mieux fixer les idées, l'exemple suivant de la figure 3.1 (purement théorique) illustre dans le domaine temporel ce que devra faire la carte audio d'un téléphone IP, dans le but de transformer le signal continu (la voix) en signal discret, ensuite quantifié, codé, compressé et envoyé sous forme de paquets à travers les différentes couches protocolaires du réseau. Dans cet exemple on multiplie le signal continu $x_c(t)$ par un train d'impulsion de Dirac [4].

3.3 Les Protocoles de communication [3]

La difficulté des protocoles utilisés en Voix sur IP (VoIP), est due à la manière de communiquer des humains. Non seulement le signal doit arriver de la même forme qu'avant sa transmission, mais il doit aussi le faire en moins de 300 millisecondes. Si des paquets sont perdus ou retardés, il y aura une dégradation de la qualité de la communication. Les protocoles de transport qui sont appelés collectivement "l'Internet" n'ont pas été conçus à l'origine pour la diffusion de média temps réel. Les terminaux traitent les paquets manquants en attendant plus longtemps jusqu'à ce qu'ils arrivent, en demandant la retransmission, ou en considérant l'information comme perdue et en poursuivant

¹Nyquist a publié deux articles "Certain Factors Affecting Telegraph Speed" (1924) et "Certain Topics in Telegraph Theory" (1928) dans lesquels il postula ce qui est connu sous le nom de théorème de Nyquist.

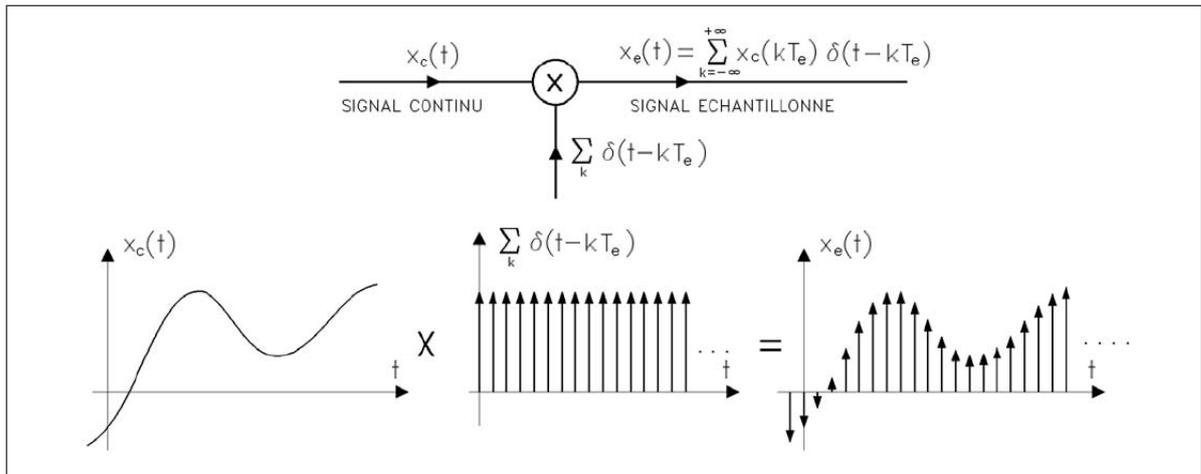


FIG. 3.1 – Échantillonnage d'un signal continu par un train d'impulsion de Dirac

sans elle. Ces mécanismes ne conviennent pas à une conversation vocale typique. Nos conversations ne s'adaptent pas bien à la perte de lettres ou de mots ou à un délai sensible entre l'émission et la réception.

Le RTC (Réseau Téléphonique Commuté) a été conçu spécifiquement pour la transmission de la voix et convient parfaitement à la tâche d'un point de vue technique. La Voix sur IP se distingue du RTC et tient la promesse d'incorporer les communications vocales dans tous les autres protocoles transportés sur le réseau. Le problème de la transmission de la voix par paquet vient de l'incompatibilité entre la façon dont nous parlons et la façon dont IP transporte les données. Parler et écouter consiste à relayer un flux audio, alors que les protocoles d'Internet sont conçus pour tout morceler, encapsuler les morceaux d'information dans des milliers de paquets et distribuer chaque paquet au destinataire. Les protocoles de VoIP constituent un pont entre Internet et la téléphonie sur IP. On décrira de ce document les protocoles de signalisation SIP, IAX et H.323. Cependant, il en existe d'autres comme MGCP, Skinny/SCCP (propriétaire Cisco) et UNISTIM (propriétaire NORTEL).

3.3.1 Le protocole SIP

SIP (Session Initiation Protocole) est un protocole de la couche application (Modèle OSI), qui utilise le port 5060 pour ces communications. Ce protocole a changé le monde de la VoIP. D'abord considéré tout au plus comme une idée intéressante, SIP semble maintenant détrôner H.323 en tant que protocole VoIP du premier choix, et certainement aux terminaisons du réseau. SIP peut être transporté par TCP ou UDP. Il est utilisé pour "établir, modifier et terminer les sessions multimédia comme les appels téléphoniques sur l'Internet" ²

SIP ne transporte pas de média entre terminaux. C'est RTP (Real Time Protocol) qui est utilisé pour transmettre le média, la voix par exemple, entre les terminaux. RTP utilise des ports non privilégiés ; dans le cas du serveur Open Source Asterisk (10 000 à 20 000, par défaut). Une topologie courante pour illustrer SIP et RTP, couramment appelée le « trapèze SIP », est montrée à la figure 3.2 . Quand Alice veut appeler Bob, le téléphone d'Alice contacte son serveur proxy, et le proxy essaye de trouver Bob (souvent en se connectant à travers son proxy). Une fois que les deux téléphones ont démarré l'appel, ils communiquent directement entre eux (si possible), pour que les données n'accaparent pas les ressources du proxy.

Le scénario de la figure 3.3 illustre un appel simple par SIP utilisant les requêtes INVITE, ACK et BYE. Un client (Alice) SIP appelle un autre terminal (Bob) en utilisant le message INVITE.

²RFC 3261, SIP, p.9 section 2

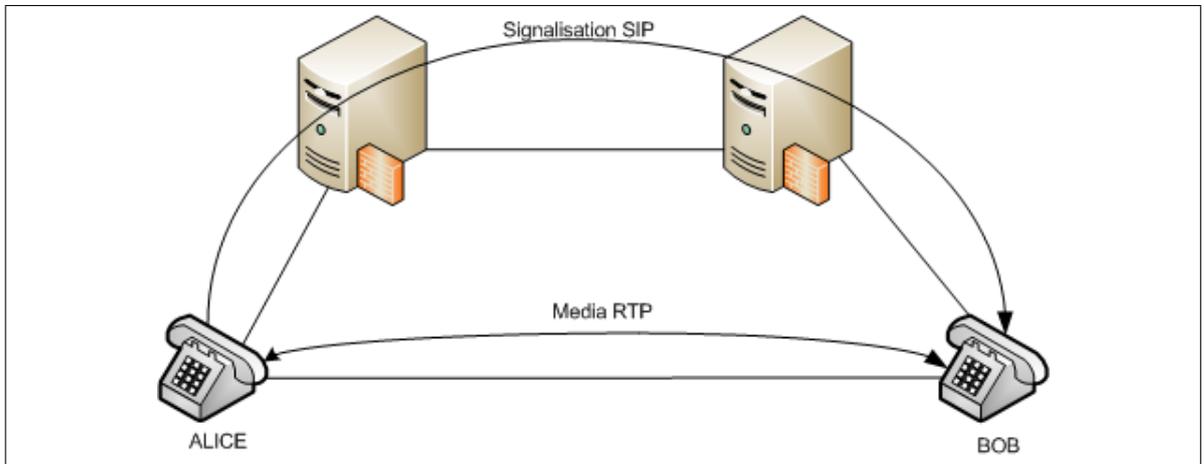


FIG. 3.2 – Le trapèze SIP

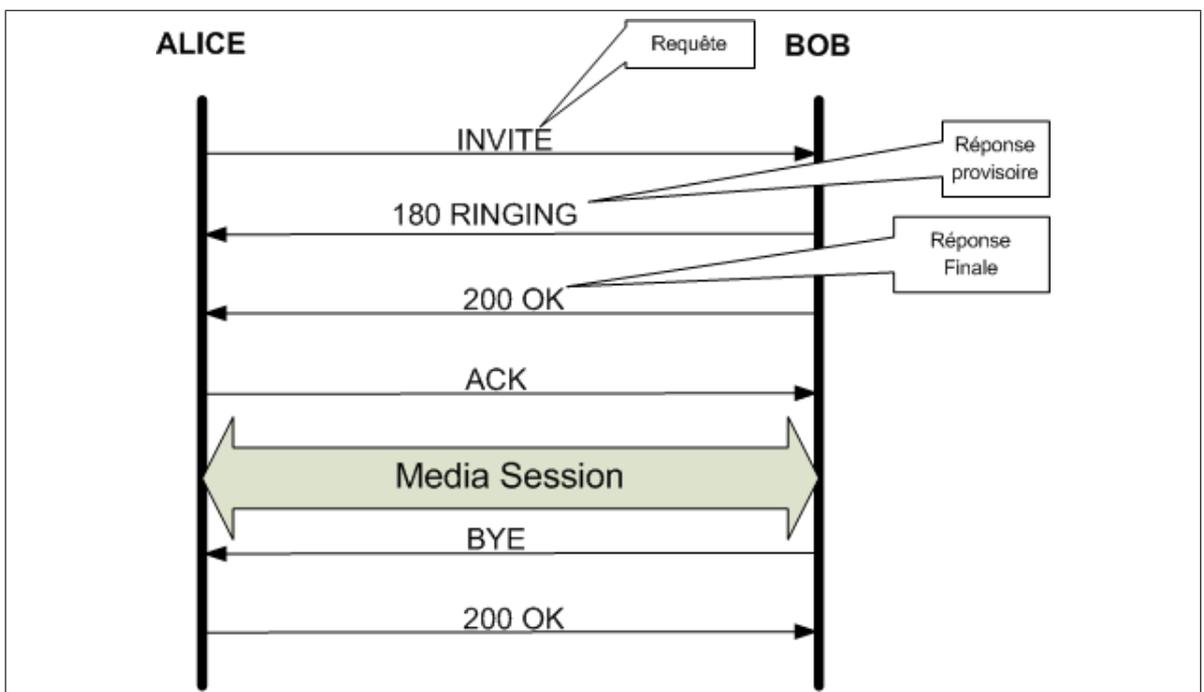


FIG. 3.3 – Scénario d'un appel simple par SIP [2]

Ce message contient d'ordinaire assez d'informations pour permettre au terminal appelé d'établir immédiatement le flux média requis vers l'appelant. Le terminal appelé doit quant à lui, indiquer qu'il accepte la requête, ce qui est la finalité de la réponse 200 OK. Puisque la requête est une invitation à communiquer, la réponse 200 OK contient également d'ordinaire une proposition de flux média pouvant être établie vers le terminal appelé, ainsi que les informations de port correspondants. Le message ACK est une transaction et la requête BYE est une demande de relâchement de l'appel. Ce type d'échange de transaction SIP pendant une communication entre user agents est appelé un dialogue SIP (dialog). Il faut préciser que la capture de ces signaux ou requêtes permet de facturer l'appel.

Sur le point de vue sécurité, SIP utilise un mécanisme de "challenge/response" pour authentifier ses utilisateurs. Un INVITE initial est envoyé au proxy auquel le terminal veut communiquer. Le Proxy envoie alors un message 407 Proxy Authorisation Request, qui contient un ensemble aléatoire de caractère appelé « nonce ». Ce nonce est utilisé avec le mot de passe pour générer un hachage MD5 qui est alors renvoyé avec l'INVITE suivant. En supposant que le hachage MD5 correspond à celui que le Proxy a généré, le client est alors authentifié.

Les attaques de type Denial of Service (DoS) sont probablement les plus communes dans les communications VoIP. Une attaque de type DoS peut se produire quand un grand nombre de requêtes INVITE invalides est envoyées au serveur Proxy pour essayer de submerger le système. SIP dispose de plusieurs méthodes pour minimiser les effets des attaques DoS mais au final elles sont impossibles à empêcher. SIP implémente une méthode TLS (Transport Layer Security) pour garantir un mécanisme de transport sécurisé. Enfin, il faut préciser que le chiffrement média (c'est-à-dire le flux RTP) est au delà de la portée de SIP et doit être géré séparément (par un mécanisme de IPsec par exemple).

3.3.2 Le protocole IAX

Le protocole IAX a été développé par Digium dans le but communiquer avec d'autres serveurs Asterisk d'où le nom « Inter-Asterisk eXchange ». IAX est un protocole de transport comme SIP qui utilise un seul port UDP/4569 tant pour le canal de contrôle que les flux RTP (Real Time Protocol). Il facilite ainsi la configuration de pare-feu et le fonctionnement derrière du NAT.

IAX inclut la capacité d'authentifier de trois façons différentes : texte clair, hachage MD5 et échange de clé RSA. Cela bien entendu, ne chiffre pas le media ou les entêtes entre les terminaux. Beaucoup de solutions incluent l'utilisation de Réseau Privé Virtuel (RPV), à l'aide de matériel ou de logiciel pour chiffrer le flux dans une autre couche de technologie, ce qui implique que les terminaux disposent de ces tunnels configurés et fonctionnels.

3.3.3 Le protocole H.323

Le protocole H.323 de l'Union Internationale des Télécommunications (UIT) a été conçu à l'origine (mai 1996) pour fournir un mécanisme de transport IP pour la vidéo-conférence. Il est devenu le standard dans les équipements de vidéo-conférence basés sur IP et il a aussi connu brièvement la gloire comme protocole VoIP. Bien que le débat pour déterminer quel protocole dominera le monde de la voix sur IP soit mouvementé, dans Asterisk, H.323 est devenu obsolète en faveur d'IAX et de SIP. H323 n'a pas eu beaucoup de succès parmi les utilisateurs et les entreprises bien qu'il soit toujours le protocole le plus utilisé chez les opérateurs.

Le standard H.323 utilise le protocole RTP de l'IETF (Internet Engineering Task Force) pour transporter le média entre terminaux.

H323 est un protocole relativement sûr et n'a pas besoin de beaucoup d'attention au-delà de celle qui est commune à tous les réseaux communicants avec l'Internet. Puisque H323 utilise le protocole RTP pour les communications média, il ne supporte pas par défaut les voies média chiffrées.

L'utilisation de RPV ou d'un autre tunnel chiffré entre terminaux est la façon la plus commune de sécuriser les communications.

3.4 La facturation avec le logiciel A2billing

Bien qu'on fasse de la commutation de paquet en téléphonie sur IP, pour des raisons économiques et pour faciliter l'implémentation des logiciels de facturation, les clients sont facturés par rapport à la durée de leurs communications.

Il faut préciser que le logiciel de facturation peut être placé ou non sur la même machine que le serveur de commutation IP. Ces deux entités communiquent au travers d'une interface pour s'échanger les signaux ou informations d'authentification d'un client, d'initialisation et de fin d'une communication...

3.4.1 L'Asterisk Gateway Interface

Pour comprendre le système de facturation A2billing il est nécessaire d'introduire le concept d'AGI "Asterisk Gateway Interface". Comme on peut déjà s'en douter le serveur IPBX "open Source" utilisé dans le cadre de ce projet est bien Asterisk dans sa version intégrée dans le paquet TRIXBOX.

L'AGI fournit une interface standardisée, qui peut être programmée en Perl, PHP et Python, pour que les programmes tiers puissent contrôler le plan de numérotation d'Asterisk. En général les scripts AGI s'utilisent pour élaborer de la logique applicative, communiquer avec les bases de données relationnelles (telles PostgreSQL ou MySQL), ou encore accéder à d'autres ressources externes.

En ce qui concerne A2billing, son script AGI a été développé par Arezqui Belaïd "Areski" en PHP. Pour plus d'informations sur la programmation des AGI en PHP voici le site de référence : <http://phpagi.sourceforge.net/> mis en ligne par Matthew Asham.

3.4.2 Les principes de base de A2billing

Lorsqu'un client SIP ou IAX authentifié sur le serveur Asterisk passe un appel il s'en suit un processus pour acheminer cet appel au destinataire suivant un algorithme bien déterminé. La figure 3.4 décrit la procédure d'un appel avec le logiciel A2billing. Par soucis de simplicité, certains détails ont été laissés. On peut retenir de façon générale l'ordre suivant des événements pour un abonné POSTPAID (facturation périodique) :

- En fonction du préfixe, A2billing recherche de la Route la moins chère par rapport à son prix d'achat ou de vente ; il s'agit du respectivement du (**Least Cost Routing / Dialing**) ;
- Test si l'abonné POSTPAID n'a pas atteint sa limitation de crédit ;
- Si le crédit le permet, l'appel est dirigé vers le destinataire et A2billing se met en attend qu'il décroche l'appel ;
- Le **RATE** (prix de vente) de la destination sera utilisé pour calculer le coût de l'appel lorsque l'appel est coupé ;
- Les informations sur le trafic du client sont mis à jour dans la base de donnée d'A2Billing et forment ce qui est appelé dans le jargon les "CDR".

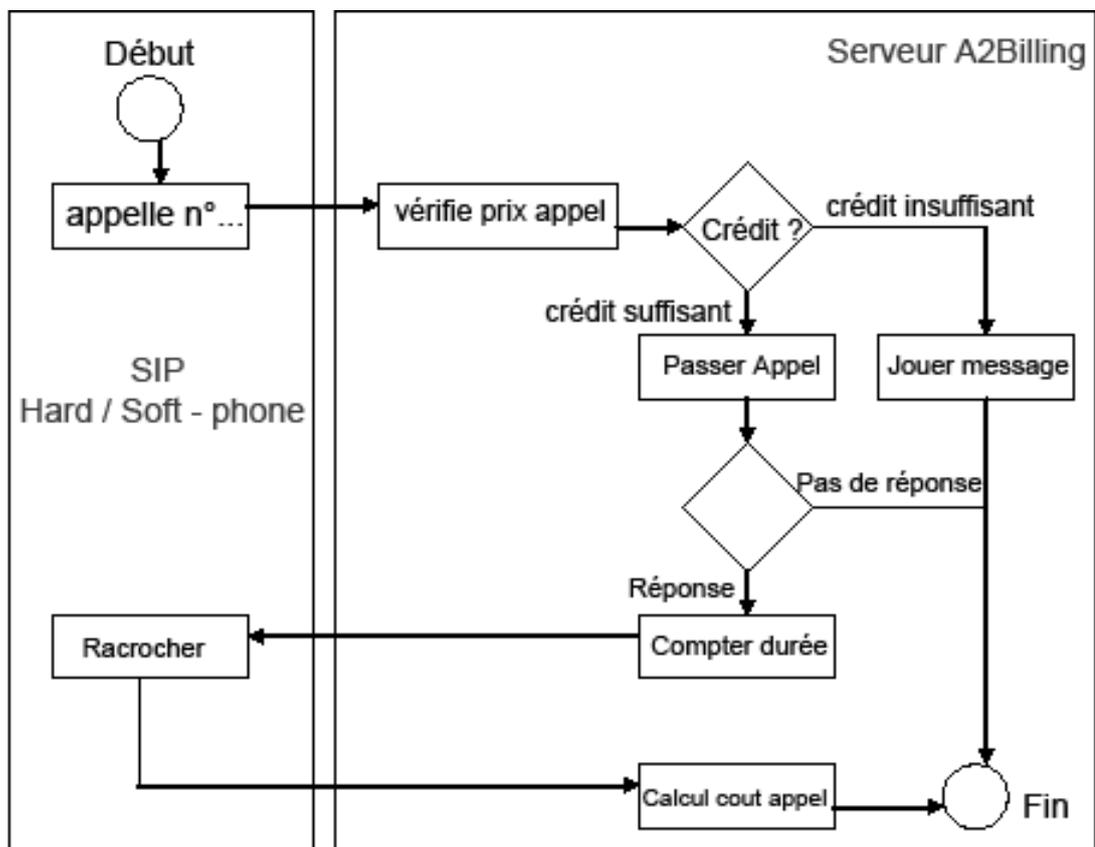


FIG. 3.4 – Procédure d'un appel A2Billing

Chapitre 4

L'environnement de test et les tests effectués

Afin de réaliser les tests précités au point 3 du cahier des charges, l'environnement décrit dans cette partie a été conçu.

4.1 L'environnement de test

L'architecture de base de l'environnement de test se présente comme suit à la figure 4.1 .

Le détail de la configuration de chaque élément se trouve à l'annexe de ce document.

Cette infrastructure mise en place permet de facturer avec le logiciel "Open Source" a2billing , les comptes client/SIP dans le réseau (Softphone ou Hardphone), pour les appels effectués vers Internet (Free World Dialup) ou via le Réseau Téléphonique Public Commuté PSTN au travers de l'interface BRI du routeur CISCO 1760 connectée entre ligne ISDN (0229400020) et le réseau LAN du laboratoire des télécommunications de l'EIG.

4.2 Résultat des tests

Conformément au cahier des charges, on a effectué dans la limite de nos possibilités, les tests dont les résultats sont les suivants :

4.2.1 Nombre d'utilisateurs dans la base

Le nombre d'utilisateur qu'on peut créer avec a2billing est illimité tant qu'il y a de l'espace sur le disque dur hébergeant la base de donnée gérée par a2billing . Le nombre d'appel simultané dépend des performances du serveur.

4.2.2 Assignation de prix par destination

A2billing supporte bien la méthode d'assignation des prix par destination. Cette méthode est détaillée dans l'annexe et le principe est expliqué au paragraphe § 3.3.2 .

4.2.3 Limitation de crédit

Par divers scénarios notamment un client atteignant sa limite de crédit en cours de communication ou une limitation à zéros franc, on a testé cette fonctionnalité. On peut donc affirmer que A2billing

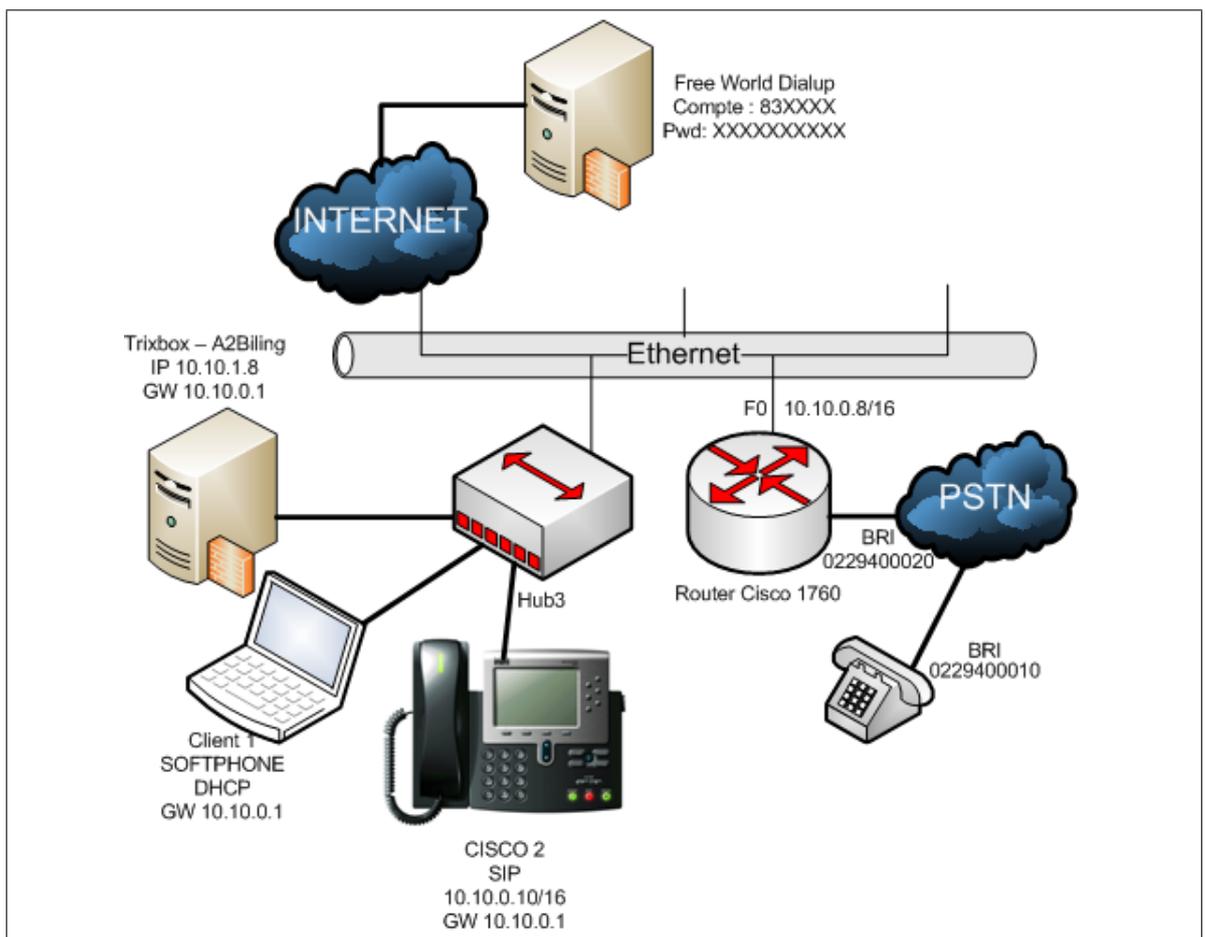


FIG. 4.1 – Environnement de test

supporte bien la méthode de limitation de crédit pour les clients POSTPAID. Le moyen d'activer cette fonction est expliqué dans l'annexe.

4.2.4 Blocage si crédit nul

Cette fonction est la conséquence du point précédent et les tests on montré que a2billing supporte comme annoncé cette fonction.

4.2.5 Interruption d'appel si le crédit est dépassé

Cette fonction a été testé en cours de communication et effectivement dès que le client n'a plus assez de crédit tout comme il atteint sa limite de crédit son appel est coupé, s'en suit alors un message avertissant que le crédit est insuffisant pour poursuivre l'appel.

4.2.6 Low Cost Route (LRC)

Cette fonction n'a pas pu être testé car l'infrastructure ne le permettait pas. Pour le faire on aurait besoin de plusieurs **PROVIDER** offrant les mêmes Routes à des prix différents.

4.2.7 Détection de la destination par préfixe

A2billing supporte bien la reconnaissance des destinations par le préfixe de l'appel sortant. L'annexe explique en détail comment faire ce paramétrage.

4.2.8 Statistiques

Les statistiques sont disponibles lorsqu'on consulte les **CALL REPORT** avec a2billing. Ce menu permet de voir aussi les statistiques sur les bénéfices réalisés ou la charge journalière, mensuelle. Pour voir à quoi ressemble l'interface graphique de A2billing sans installer tout l'environnement, il faut visiter le lien suivant : (<http://demo.asterisk2billing.org/ndemo/A2BillingUI/Public/index.php>) . Puis s'enregistrer avec Username *demo*, Password *demo*.

4.2.9 Méthodes de paiement

Cette fonction n'a pas pu être testé car l'infrastructure ne le permettait pas. Pour le faire on aurait eu besoin par exemple d'un compte Paypal via une carte de crédit. Cependant s'il y a un réel intérêt pour ce test et si les moyens sont mis à disposition, il pourra être fait.

Chapitre 5

Étude économique succincte du projet

A cette étape du projet, après avoir répété plusieurs fois le terme "Open Source", on se pose tous finalement la question combien ça coûte et qu'est ce ça rapporte de déployer un système Trixbox-A2billing .

5.1 Coûts de déploiement

Le coût pour le déploiement du système Trixbox-A2billing dépend du nombre d'extension géré dans l'environnement ToIP. A titre indicatif voici ce qu'il faut prévoir pour le serveur Trixbox-A2billing :

- 25 postes : 900CHF
- 250 postes : 5000CHF
- 1000 postes : 2 serveurs avec load-balancing (15000CHF)

Un téléphone Ip comme le "IP Phone BudgeTone-101", proposé sur le site de SWITZERNET coûte 92.- CHF.

Il faut prévoir Réseau 100 Mbits "switché" avec segmentation en Vlan et Qualité de service (QoS)

La gateway Cisco 1760 pour la connexion vers le PSTN coûte actuellement (juin 2007) environ 1600.-CHF.

Le Trixbox, le logiciel de facturation A2billing et le Softphone X-lite sont entièrement gratuit. Cependant un don pour l'équipe ayant développé le système serait une bonne idée.

5.2 Comparaison avec une solution payante

Le gain en déployant un système "Open Source" par rapport à une solution payante est de 100 %. Le coût du matériel est le même, mais dans le cas de la solution payante, il faut ajouter le prix des divers licences, à savoir, la licence par rapport au nombre d'utilisateurs, la licence du logiciel de facturation et éventuellement, la licence du système d'exploitation sur lequel tourne le serveur ToIP. Les tableaux 5.1¹ et 5.2 illustre respectivement le coût d'une licence pour le "Callmanager 4.2" de Cisco et le logiciel de facturation payant "Nextone".

¹Source fiche technique CISCO UNIFIED CALLMANAGER VERSION 4.2

Modèle de serveur	Référence	Nombre maximal de téléphones par serveur	Prix (US\$)
Cisco MCS-7815-I1	CM4.2-K9-7815SE	100	1995
Cisco MCS-7815-I1	CM4.2-K9-7815-I1	300	3995
Cisco MCS-7815-I1 redondant	CM4.2-K9-7815R	300	6995
Cisco MCS-7825-H1	CM4.2-K9-7825-H1	1 000	5995
Cisco MCS-7825-I1	CM4.2-K9-7825-I1	1 000	5995
Cisco MCS-7835-H1	CM4.2-K9-7835-H1	2 500	7995
Cisco MCS-7835-I1	CM4.2-K9-7835-I1	2 500	7995
Cisco MCS-7845-H1	CM4.2-K9-7845-H1	5 000	15995
Cisco MCS-7845-I1	CM4.2-K9-7845-I1	5 000	15995
HP DL320-G3	CM4.2-K9-DL320	1 000	5995
HP DL380-G4, un seul processeur	CM4.2-K9-DL380	2 500	7995
HP DL380-G4, deux processeurs	CM4.2-K9-DL380D	5 000	15995
IBM x306 3.4-GHz Serveur	CM4.2-K9-X306	1 000	5995
IBM X346, un seul processeur	CM4.2-K9-X346	2 500	7995
IBM X346, deux processeurs	CM4.2-K9-X346D	5 000	15995
IBM x206 Serveur	CM4.2-K9-X206	300	3995

TAB. 5.1 – Prix licence par nombre d'utilisateur Callmanger 4.2

DESCRIPTION	DURÉE	FRAIS
• Installation et formation		\$2.000
• Licence 300 ports	12 mois	\$3.000/mois
• Achat après 12 mois	Illimitée	\$10.000

TAB. 5.2 – Prix licence logiciel Nextone

Chapitre 6

Conclusion

Il est possible à l'échelle d'une PME de déployer un système VoIP "Open Source" intégrant la facturation des clients. Cependant, il faut prendre soin de sécuriser le réseau. La première recommandation est de séparer le réseau voix du réseau données en créant deux Vlan distincts.

L'utilisation des Softphones n'est pas recommandée pour avoir un minimum de sécurité, en ce sens que le pirate peut profiter des failles du système d'exploitation sur lequel est installé le softphone.

Il faudrait aussi sécuriser les accès à distance sur les serveurs. Pour rester dans la philosophie de l' "Open Source", on pourrait proposer FreeRADIUS [5] et verrouiller l'accès au serveur par identification des adresses MAC des clients distants. Compte tenu du temps imparti pour ce travail, cet aspect n'a pas été développé dans ce rapport.

Une des faiblesses du système testé est qu'il ne répond pas au critère de haute disponibilité permettant un démarrage à chaud avec une architecture redondante. Ce critère est crucial dans des environnements sérieux où la non disponibilité du système peut entraîner des risques et des pertes très élevés. C'est ce qu'on ne pourra, par exemple, pas admettre dans un environnement bancaire.

A ce jour ce qui empêche les systèmes VoIP de remplacer comme annoncé les PABX classiques c'est la qualité de service (QoS). Comme suite intéressante à ce travail on pourrait se pencher sur cette problématique de la qualité de service. On peut donc ouvrir une passionnante thématique de recherche pour voir l'incidence des différents composants d'une infrastructure voix sur IP sur la qualité de service. Par exemple, une meilleure négociation des "Codecs" en fonction de la bande passante peut faire gagner en qualité du signal transmis. Une série de mesures réelles à effectuer sur le réseau IP peut permettre de donner plus de crédibilité à cette assertion.

On pourrait alors rechercher des modèles mathématiques qui simuleront cette réalité et aideront à mieux dimensionner les réseaux VoIP.

Chapitre 7

Remerciements

J'adresse mes remerciements à Mme Sona Gabrielyan de Switzernet pour m'avoir proposé ce thème de travail.

Je remercie également mes professeurs, Messieurs, Andrès Revuelta, Tewfiq El Maliki, Gerald Litzistorf et Eric Yenni pour leur disponibilité, et pour m'avoir aidé et orienté sur mes questions.

Je n'oublie pas l'assistant Nicolas Sadeg.

Un remerciement particulier à tous mes collègues du laboratoire de Télécom pour m'avoir permis de les appeler plusieurs fois lors de mes premiers tests.

Enfin tous ceux qui de près ou de loin m'ont aidé dans ce travail je vous en suis reconnaissant.

Annexe A

Mise en place de l'environnement de test

L'environnement de test présentée à la figure 4.1 est constitué par une infrastructure matérielle et une infrastructure logicielle

A.1 Infrastructure matérielle de test

La plate forme matérielle de test comprend :

- 2 PC clients XP sur lesquels tourne un softphone X-lite (téléchargeable gratuitement à l'adresse <http://www.counterpath.com/>);
- 1 PC dédié en Serveur IPBX *Performance 512M de RAM Pentium 4*
- 1 HUB 4 Ports
- 1 Hardphone Cisco 7960
- 1 Router Cisco 1760 IOS version 12.2

A.2 Infrastructure logicielle de test

La plate forme logiciel de test comprend :

- Le softswitch IPBX TrixBos
- Des Softphones X-LITE
- Le Logiciel open source de facturation A2BILLING

A.3 Installation du IPBX TrixBos

TrixBos est très facile à installer et à manipuler. Il permet en quelques minutes, de mettre en place un système complet de Toip (Téléphonie sur IP). Le site Web de TrixBos est : <http://www.trixbox.org>. Après avoir téléchargé l'image ISO du logiciel, il faut le graver sur un CD ROM. Par la suite il faudra « bootter » le PC dédié au serveur IPBX, sur le CD gravé. Pas besoin d'être un Linux Guru pour faire l'installation, il suffit juste de suivre les étapes comme présentés dans un magnifique tutoriel disponible gratuitement à l'adresse : <http://dumbme.voipeye.com.au/trixbox/>

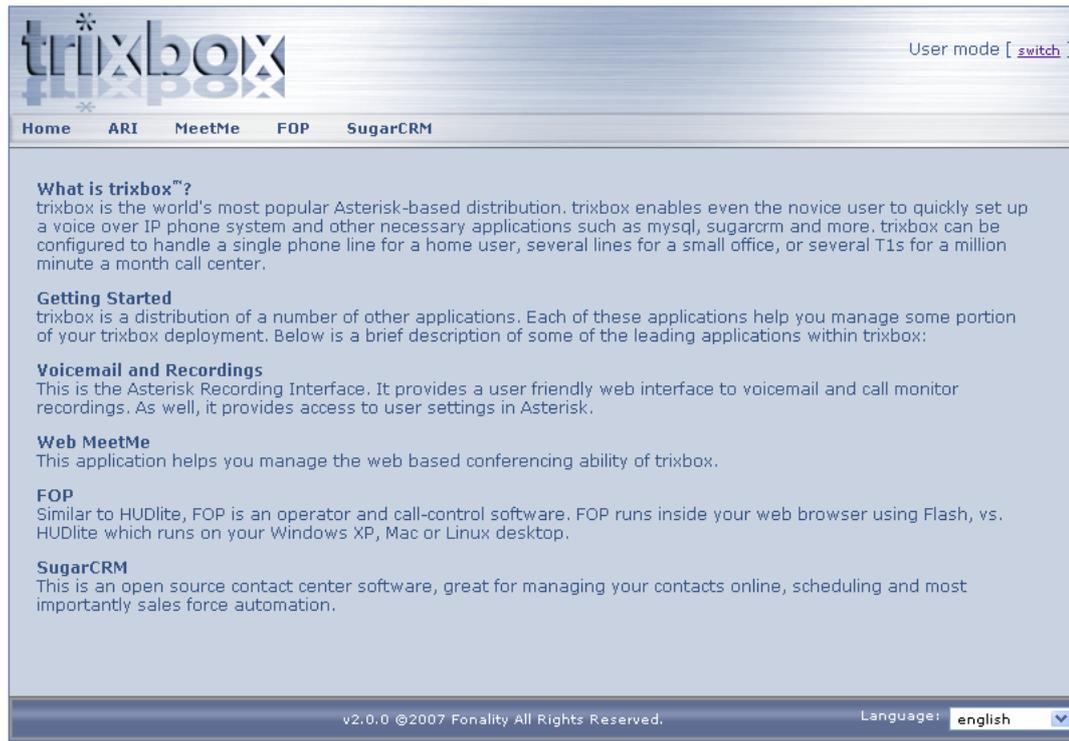


FIG. A.1 – Page d'accueil TrixB0x

A.4 Configuration du TrixB0x [1]

Les fonctions disponibles dans le TrixB0x sont nombreuses et intéressantes, cependant on se limitera dans cette partie du rapport à la configuration des "TRUNK" permettant de communiquer avec l'extérieur à partir du serveur TrixB0x.

La partie centrale du travail est la configuration des comptes SIP qui seront facturés, cette partie sera traitée plus loin avec le logiciel de facturation car, c'est ce logiciel qui crée et gère ces propres comptes SIP. Dès lors, il est important de préciser que les communications de tous les comptes créés directement via l'interface Web du TrixB0x, ne sont pas facturées.

Après l'installation du TrixB0x, il faut se connecter via une machine du réseau, à l'interface web du TrixB0x, en saisissant dans un navigateur au choix, l'adresse IP du serveur TrixB0x. Dans le cas de ce projet le serveur TrixB0x se trouve à l'adresse `http://10.10.1.8`. On arrive sur une interface qui se présentant comme suit à la figure A.1 .

En cliquant sur le lien *switch* (en haut à droite) on arrive à l'invite d'authentification de la figure A.2 .

Le login par défaut pour s'authentifier est *maint* et le mot de passe *password*.

Les liens *Asterisk* puis *FreePBX* du menu conduisent à la page de figure A.3

Ici on cliquera sur *Tools* . On obtient l'image de la figure A.4 .

Puis sur *Module – Admin* et on obtient l'image de la figure A.5 .

C'est dans cette partie qu'il faut sélectionner les modules qui seront utilisé par la suite. On a coché puis validé : Core, Feature Code Admin, Time Conditions, Voicemail, On Hold Music, IVR, Queues, Recordings and Backup & Restore Pour plus de détails sur cette partie, il existe une bonne documentation à l'adresse : `http://www.sureteq.com/asterisk/trixboxv2.0beta.htm`

Pour poursuivre la configuration de notre environnement de test, on va créer un compte IAX gratuit, qui permettra de sortir des appels tests du réseau local vers Internet. Rendez-vous donc à l'adresse `http://freeworlddialup.com` ou il faut s'enregistrer et créer un compte IAX gratuitement.



FIG. A.2 – Invite d'authentification

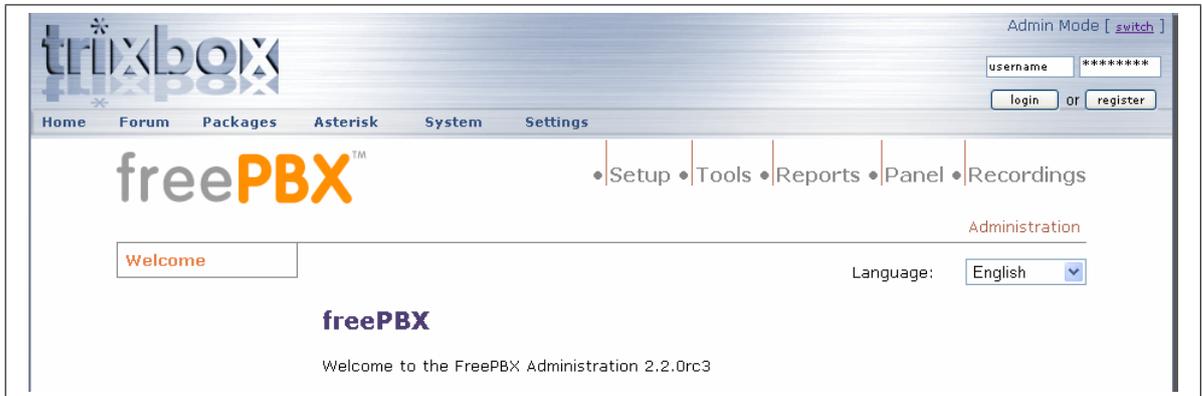


FIG. A.3 – Interface FreePBX



FIG. A.4 – Tools de FreePBX

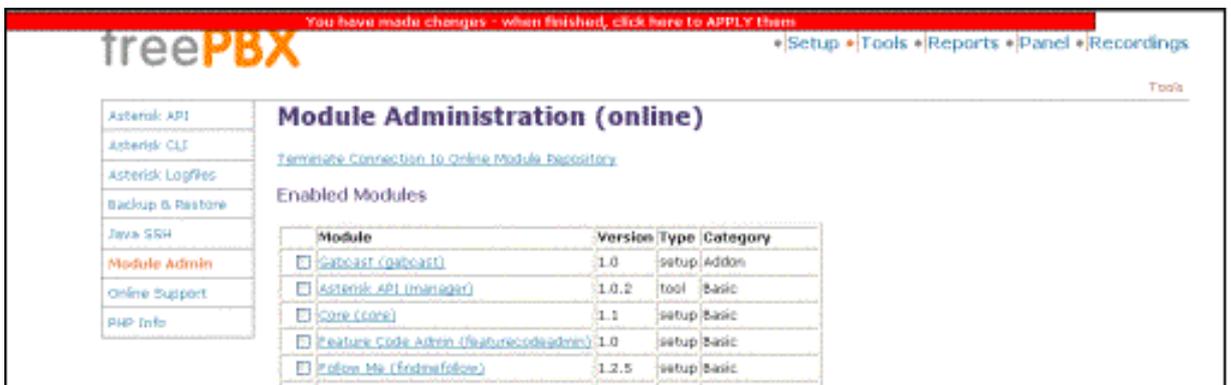


FIG. A.5 – Module Admin de Tools

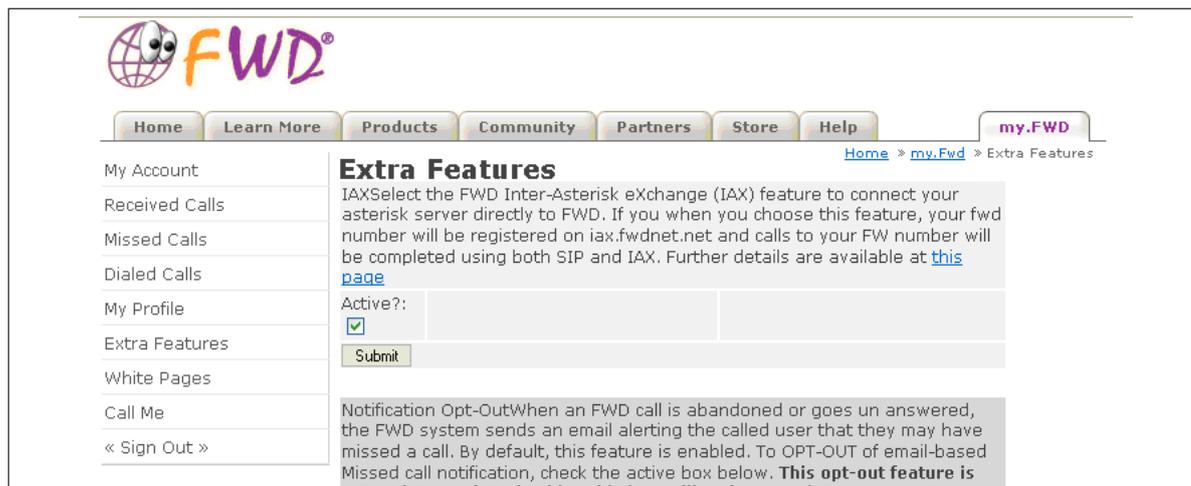


FIG. A.6 – Activation du compte IAX sur Free World Dialup

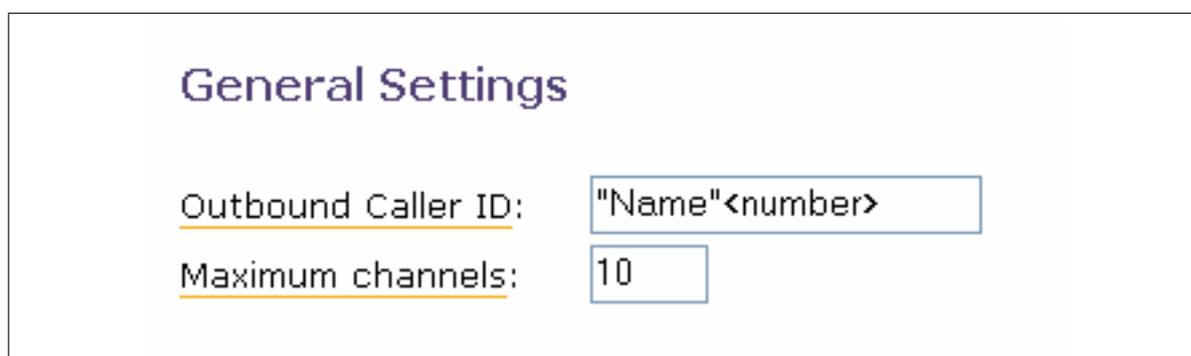


FIG. A.7 – Name et Number Trunk

Ne surtout pas oublier d'activer l'option IAX dans le menu *Extra Feature* comme indiqu      la figure A.6 . L'activation du compte n  cessite environ 10 minutes.

Apr  s cette   tape, on retourne vers FreePBX de TrixB0x pour configurer le *Trunk*. Pour ce faire, on click sur *Setup* dan le menu en haut puis *Trunks* dans le menu de gauche de la page qui appara  t, ensuite sur *Add IAX Trunk*. On remplacera *name* et *number* par les param  tres du compte IAX cr    chez freeworlddialup comme sur la figure A.7.

Pas besoin de configurer la section *Outgoing Dial Rules* de la figure A.8 .

Cependant, les sections *Outgoing Settings*, *Incoming settings* et *Registration*, n  cessitent certains param  trages. Voici les param  tres aux figure A.9, A.10 et A.11, les param  tres    saisir.

Une remarque g  n  rale et importante est qu'   chaque soumission d'une nouvelle configuration, il faut cliquer sur la barre horizontale en rouge en dessous du menu pour recharger le syst  me.

Maintenant, on a besoin de configurer par la suite une Route, pour que les appels destin  s    ce Trunk puissent l'emprunter. On click alors sur *Outbound Route* et on saisi les param  tres suivant comme    la figure A.12 .

On expliquera plus loin, au moment venu, pourquoi le param  tre Dial Patterns est mis    393 sur la figure A.12 . Il faut reprendre la m  me proc  dure pour cr  er la Route vers le PSTN    travers l'interface BRI. La configuration de TrixB0x est ainsi termin  e pour les fonctions minimales n  cessaires pour les tests, on passe    celle du logiciel de facturation A2Billing.

Outgoing Dial Rules

Dial Rules:

Dial rules wizards:

Outbound Dial Prefix:

FIG. A.8 – Outgoing Dial Rules

Outgoing Settings

Trunk Name:

PEER Details:

```
host=iax2.fwdnet.net
secret=projet2007
type=peer
username=833060
```

FIG. A.9 – Outgoing Settings

Incoming Settings

USER Context:

USER Details:

```
allow=ulaw
auth=md5
context=from-pstn
disallow=all
secret=projet2007
type=user
```

FIG. A.10 – Incoming settings

Registration

Register String:

FIG. A.11 – Registration

Add Route

Route Name:

Route Password:

Emergency Dialing:

Dial Patterns

Insert: ▾

Trunk Sequence

▾

FIG. A.12 – Registration

A.5 Installation de A2Billing

Il existe plusieurs manières pour installer A2Billing, Voici celle qu'on propose :

1. Télécharger `Asterisk2Billing_release_Chameleon_v1.2.3.tar.gz` à l'adresse `http://www.asterisk2billing.org` sur un PC client XP se situant dans le même réseau que le serveur Asterisk ;
2. Décompresser le fichier ainsi télécharger dans un dossier sur votre disque local
3. Connectez-vous par mode ssh (de préférence graphique) à la machine sur laquelle tourne le serveur Asterisk.
4. Créer un dossier `a2billing` sur la machine Asterisk dans le repertoire `/usr/src/`
5. Copier le dossier `Trunk` obtenu lors de la décompression de A2Billing et placer le dans le dossier `A2Billing` de la machine TrixBos
6. Passer en mode CLI sur la machine TrixBos puis saisissez les commandes suivantes :


```
cd /usr/src/a2billing/trunk/DataBase/mysql/Mysql-3.x.4.x
mysql -u root -ppassw0rd < a2billing-MYSQL-createdb-user.sql
mysql -u root -ppassw0rd mya2billing < a2billing-mysql-schema-MYSQL.3.X-4.X.v1.2.3.sql
```
7. Toujours en ligne de commande, on installe le HTTP Interface :


```
cd /var/www/html
mv a2billing a2billing_old
mkdir a2billing
chown -R asterisk :asterisk a2billing
cd a2billing
cp -r /usr/src/a2billing/trunk/A2Billing_UI / . .
cd /var/www/html
mv a2customer a2customer_old
mkdir a2customer
chown -R asterisk :asterisk a2customer
cd a2customer
cp -r /usr/src/a2billing/trunk/A2BCustomer_UI / . .
```
8. Installation de l'AGI


```
cd /var/lib/asterisk/agi-bin
cp /usr/src/a2billing/trunk/A2Billing_AGI/a2billing.php .
cp -r /usr/src/a2billing/trunk/A2Billing_AGI/libs_a2billing .
chown asterisk :asterisk a2billing.php
chown -R asterisk :asterisk libs_a2billing
```
9. Installation de A2Billing.conf


```
cd /etc/asterisk
mv a2billing.conf a2billing_old.conf
cp /usr/src/a2billing/trunk/a2billing.conf .
chown asterisk :asterisk a2billing.conf
```
10. Changez la base de donnée de type postgres en mysql


```
nano /etc/asterisk/a2billing.conf
```

Dans la section [database] on change

```
dbtype=postgres
```

```
;dbtype=mysql
```

En :

```
;dbtype=postgres
```

```
dbtype=mysql
```

Puis CTRL+X et Y suivi de Enter pour confirmer et quitter l'éditeur nano

11. Installation des fichiers sons

En mode ssh, de préférence graphique, copiez tous les fichiers contenu dans le dossier sound dans le repertoire /trunk/A2Billing_AGI/sounds et placez les dans le dossier /var/lib/asterisk/sounds/ du serveur Asterisk

12. Configuration du manager

Editer le fichier /etc/asterisk/manager.conf et vérifier les paramètres suivants :

```
[general]
```

```
enabled = yes
```

```
port = 5038
```

```
bindaddr = 0.0.0.0;displayconnects = yes
```

Ajouter à manager.conf les lignes suivantes :

```
[myasterisk]
```

```
secret = mycode
```

```
read = system,call,log,verbose,command,agent,user
```

```
write = system,call,log,verbose,command,agent,user
```

13. Inclure les clients SIP et IAX de A2Billing dans Asterisk

Dans le fichier /etc/asterisk/sip.conf, ajouter la ligne suivante :

```
#include additional_a2billing_sip.conf
```

Dans le fichier /etc/asterisk/iax.conf, ajouter la ligne suivante

```
#include additional_a2billing_iax.conf
```

14. Dire à Asterisk que faire lorsqu'un client A2Billing (context = a2billing) passe un appel

Editer le fichier /etc/asterisk/extensions.conf et insérer les lignes suivantes :

```
[a2billing]
```

```
; CallingCard application
```

```
exten => _X.,1,Answer
```

```
exten => _X.,2,Wait,2
```

```
exten => _X.,3,DeadAGI,a2billing.php
```

```
exten => _X.,4,Wait,2
```

```
exten => _X.,5,Hangup
```

15. Maintenant, testez votre installation en ouvrant dans un navigateur l'URL `http://<asterisk ip>/a2billing`, par défaut User = `admin` password = `mypassword`, On devrait avoir quelque chose qui ressemble à l'interface de la figure A.13 .

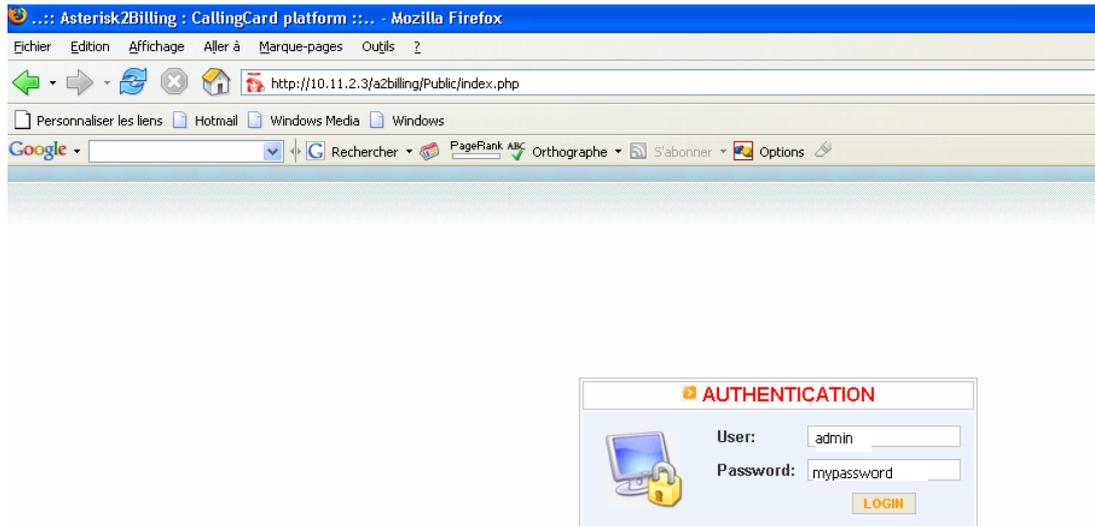


FIG. A.13 – Installation réussie

A.6 Configuration de A2Billing

Il est fortement recommandé de changer le nom d'utilisateur et le mot de passe, dès la première connexion, passez donc à *Administrator*, *Show Administrator* dans le menu de gauche, puis *Edit* pour changer les paramètres des comptes par défauts. Ici, dans ce projet on se place dans le contexte d'une société qui gère des clients POSTPAID. Lorsqu'un nouveau client veut s'enregistrer, il faut lui créer un compte SIP depuis A2Billing. Pour ce faire, A2Billing, doit avoir des PROVIDERS, vers lesquels il redirige et facture les appels, suivant les indicatifs des numéros saisis par le client. On utilisera, pour les tests, le compte freeworlddialup et trunk de la gateway cisco créé précédemment. Ce choix parce que, ce provider offre des numéros de tests gratuit comme par exemple 612 pour écouter la météo, 613 pour l'écho.... On configurera "freeworlddialup" comme trunk de sorti pour tous les numeros commençant par 393. "393", parce que lors de la configuration du trunk "fwd" dans TrixBos, si vous vous en souvenez, on avait spécifié que le "Dial Pattern = 393" cela veut dire 393 suivi de n'importe quel suite de nombre. Une opération similaire a été faite pour le trunk de sortie vers le PSTN pour tous les numéros commençant par 0.

Ceci étant dit, voici comment configurer le *trunk* de free world dialup, puis mettre les tarifs et activer des clients :

1. On click sur *TRUNK* dans le menu de gauche de A2Billing, puis sur *Create Provider*. Voici les informations à saisir :
 - Provider name = freeworlddialup**
 - Description = Mettez tout ce que vous voulez**
 - Ensuite cliquez sur confirm data**
2. Cliquer ensuite sur *Add Trunk* et valider les paramètres suivants :
 - VoIP-Provider : freeworlddialup**
 - Label : myfwd**
 - Add Prefix : (vide)**
 - Remove Prefix : 393**
 - Provider Tech : IAX2**
 - Provider IP : fwd**

TARIFFGROUPNAME	<input type="text" value="tarrifgroupe1"/>										
LC TYPE	LCR : According to the buyer price ▾										
REMOVE INTER PREFIX	Yes <input type="radio"/> - No <input checked="" type="radio"/> Remove the international access prefix (00 or 011) before matching the dialled digits with the rate 0044 for a call to the UK, only 44 would be delivered.										
RATECARD	<table border="1"> <thead> <tr> <th colspan="2">RATECARD LIST</th> </tr> </thead> <tbody> <tr> <td colspan="2" style="text-align: center;">NoRATECARD</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th colspan="2">ADD A NEW RATECARD</th> </tr> </thead> <tbody> <tr> <td>RATECARD</td> <td>1 - (default) ▾</td> </tr> <tr> <td colspan="2" style="text-align: center;"> <input type="button" value="Add"/> </td> </tr> </tbody> </table>	RATECARD LIST		NoRATECARD		ADD A NEW RATECARD		RATECARD	1 - (default) ▾	<input type="button" value="Add"/>	
RATECARD LIST											
NoRATECARD											
ADD A NEW RATECARD											
RATECARD	1 - (default) ▾										
<input type="button" value="Add"/>											

FIG. A.14 – Rajouter une Ratecard à un Tarifgroupe

Additional Parameter : (vide)

Failover Trunk : (not defined)

- Créez votre première rate card en cliquant sur *RATECARD*
Cliquez sur *Create new RateCard* et saisissez :
Tariffname : default Start date : (laisser par défaut) Expiry date : (laisser par défaut)
Trunk : myfwd Description : Tous ce que vous voulez DNID Prefix : all
- Confirmez puis cliquez sur *Add rate* puis entrer et confirmer les paramètres suivants : **Ratecard : default Dial prefix : defaultprefix Destination : default Buying rate : 0.02 Buyrate Min duration : (vide) Buyrate Billing block : (vide) Selling rate : 0.05 Sellrate Min duration : 60 Sellrate billing block : (vide) Connect charge : (vide) Disconnect charge : (vide) Start date : (laisser par défaut) Stop date : (laisser par défaut) Start time : (laisser par défaut) End time : (laisser par défaut) Trunk : myfwd**
- Cliquez sur *Ceate TariffGroup* puis entrer les paramètres suivants : **Tariffgroupname : default LC Type : LCD (save my customers the money) Remove Inter Prefix : No**
- Cliquer ensuite sur *confirm data* et par la suite sur *edit* sur la ligne du *TariffGroup* que vous venez de configurer
- Dans la section Add New Ratecard de la page, cliquez sur le bouton Add et confirmer comme sur la figure A.14 .
- Maintenant il faut créer les clients, cliquez alors successivement sur *Customers*, *textitCreate Customers*, puis rentrer les paramètres caractéristiques de votre client :
Card number : (leave default...should be a 10 digit number) Card alias : (leave default)
WebUI password : (you can leave this default, or change it to something else...it is t he customer's password to get into the a2customer web interface)
Balance : 0.00
Language : (leave default)
Tariffgroup : default
DIDGroup : Not Defined
Campaign : Not Defined
Callback : (blank)
Activated : Yes
Signup confirmation : Yes
Simultaneous Access : Simultaneous Access (this allows for the card to be in use more than once at a time)

Currency : (leave default)
Run Service : Yes
Run Autorefill : No
Initial balance : 0
Card type : Postpay card
Credit limit : 50
Enable Expiry : No Expiry
Expiry Date : (leave default)
Expiry Days : 0
VAT : (leave blank...unless you pay Value Added Tax...it is measured in percentage...so a 10 here means 10 percent)
Last name : (customer's last name)
First name : (customer's first name)
Email : (customer's email address)
Address : (customer's address)
City : (customer's city)
State/Province : (customer's state/province)
Country : (customer's country)
Zip/Postal code : (customer's Zip/postal code)
Phone number : (customer's phone number)
Fax number : (customer's fax number)
SIP account : Yes
IAX account : Yes
In use : (leave default)

9. Après avoir confirmé, cliquez sur le bouton SIP figurant sur la ligne du nouveau client créé. Editer les paramètres du client (définir par exemple l'option NAT, canreinvite...), puis cliquer sur `Generate_additional_a2billing_sip.conf`
10. Une dernière étape qui peut être très utile est d'ouvrir le fichier `/etc/asterisk/a2billing.conf` et de changer les paramètres suivant le contexte dans lequel on déploie le système. ce fichier bien commenté est assez intuitif pour sa configuration.

A.7 Configuration des Softphones

Après la première extension créée avec A2Billing, il faut configurer le client (téléphone) qui utilisera le compte SIP. Pour ce faire :

1. Télécharger le client softphone X-Lite 3.0 à l'adresse <http://www.counterpath.com>
2. Installer et lancer X-Lite,
3. Configurer le compte SIP avec les paramètres du compte que vous avez créé précédemment, vous pouvez utiliser la documentation suivante pour vous aider : http://www.counterpath.com/docs/X-Lite3.0_UserGuide.pdf, page 4
4. Composer le 393613 (Ce numéro vous permet de faire un test d'écho depuis votre compte freeworldialup) L'environnement est maintenant prêt et tous les appels passés depuis le client X-Lite sont facturés par A2Billing.

A.8 Detail et facture des appels

Pour voir les details des appels en tant que Administrateur il faut aller sur *CDR Report* dans le menu gauche de l'interface web admin de a2billing. Puis saisir les paramètres de date ou clients pour affiner la recherche

Les clients ont aussi accès à leur propre trafic par une interface Web User.

A.9 Configuration du HardPhone Cisco 7960

Le detail de cette opération se retrouve dans le document "Converting a Cisco 7940/7960 SCCP to a SIP Phone and the Reverse Process" disponible sur le site CISCO www.cisco.com

A.10 Configuration du Routeur Cisco 1760

Voici le code source de la configuration du routeur Cisco 1760

```
Current configuration : 1976 bytes
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
hostname Router
!
logging queue-limit 100
!
tdm clock bri-auto
ip subnet-zero
!
!
ip tcp path-mtu-discovery
ip name-server 10.10.1.10
!
!
template mon
!
!
isdn switch-type basic-net3
isdn voice-call-failure 0
!
voice call send-alert
voice call carrier capacity active
voice rtp send-recv
!
voice service pots
!
voice service voip
signaling forward unconditional
sip
!
voice class codec 1
```

```
codec preference 1 g711alaw
!
!
!
no voice hpi capture buffer
no voice hpi capture destination
!
!
mta receive maximum-recipients 0
!
!
!
  interface FastEthernet0/0
ip address 10.10.0.8 255.255.0.0
no ip mroute-cache
load-interval 60
speed auto
full-duplex
no cdp enable
ip rsvp bandwidth 1 1
!
interface FastEthernet0/0.1
!
  interface BRI0/0
no ip address
isdn switch-type basic-net3
isdn incoming-voice voice
isdn sending-complete
isdn skipsend-idverify
!
interface BRI0/1
no ip address
isdn switch-type basic-net3
!
  ip default-gateway 10.10.0.1
ip classless
ip route 0.0.0.0 0.0.0.0 10.10.0.1
ip route 10.11.0.0 255.255.0.0 10.10.0.1
no ip http server
!
!
!
!
call rsvp-sync
!
  voice-port 0/0
input gain 8
compand-type a-law
cptone CH
connection plar 9400020
!
voice-port 0/1
cptone CH
```

```
!  
!  
mgcp profile default  
!  
dial-peer cor custom  
!  
!  
!  
    dial-peer voice 9400020 voip  
    huntstop  
    application session  
    destination-pattern ...  
    rtp payload-type nte 98  
    voice-class codec 1  
    session protocol sipv2  
    session target ipv4 :10.10.1.8  
    dtmf-relay cisco-rtp  
    signal-type ext-signal  
    !  
    dial-peer voice 1 pots  
    huntstop  
    application session  
    destination-pattern 0T  
    port 0/0  
    forward-digits all  
    !  
    gateway  
    timer receive-rtcp 5  
    !  
    sip-ua  
    max-forwards 15  
    timers trying 1000  
    timers expires 300000  
    timers notify 1000  
    sip-server ipv4 :10.10.1.8  
    !  
    !  
    line con 0  
    exec-timeout 0 0  
    line aux 0  
    line vty 0 5  
    no login  
    !  
End
```

Bibliographie

- [1] Barrie Dempster and Kerry Garrison. *TrixBos Made easy*. Packt Publishing, Birmingham - Mumbai, 2006.
- [2] Olivier Hersent David Gurle et Jean-Pierre Petit. *L'essentiel de la VoIP*. DUNOD, Paris, 2005.
- [3] Jim Van Meggelen Jared Smith et Leif Madsen Traduction de Alexandre Belloni et Yann Serra. *Asterisk La téléphonie Open Source*. O'Reilly, 2006.
- [4] Daniel Dubois *Modification* Michel Kocher et Maurizio Tognolini. *Support de cours Traitement numérique du signal*. École d'Ingénieurs de Genève, Genève, 2006.
- [5] Jonathan Hassell. *RADUIS Securing Public Access to Private Ressources*. O'Reilly, second edition, 2003.