2. ASTERISK

In questo capitolo viene descritto Asterisk, il gateway VoIP PBX utilizzato in questa tesi per testare semplici funzionalità da centralino telefonico per la gestione di chiamate e videochiamate tra due o più computer tramite rete IP.

» 2.1 Introduzione ad Asterisk

Asterisk è un software open source sviluppato dalla Digium (www.digium.com) in ambiente Linux che permette di realizzare a basso costo una soluzione completa di PBX voice over ip, ossia una vera e propria centralina telefonica per uso privato.

La prima stesura venne realizzata da Mark Spencer che attorno all'anno 2000 fondò Digium, una società di produzione elettronica, e per favorire la diffusione dei propri prodotti fece sviluppare un'applicazione in grado di attribuire ad un PC, equipaggiato di interfaccie Digium, le funzionalità tipiche di un centralino telefonico.

Il suo nome, Asterisk, proviene dal mondo Unix e Dos dove rappresenta un cosiddetto "carattere jolly" (*) cioè la possibilità di rappresentare ogni file.

In modo simile, Asterisk è stato progettato per interfacciare qualsiasi tipo di apparato telefonico standard (sia hardware che software) con qualsiasi applicazione telefonica standard, in modo semplice e consistente. Essendo Asterisk un progetto Open Source, esso è rilasciato sotto licenza GNU GPL ed è possibile per tutti accedere al suo contenuto liberamente.

Asterisk supporta tre protocolli VoIP: IAX (sviluppato specificatamente per Asterisk stesso), SIP e H.323.

In particolare, Asterisk è un sistema ibrido poiché utilizza sia la tradizionale tecnologia TDM che i protocolli del packet voice (Voice over IP e Voice over Frame Relay).

Esso si comporta come un PBX completo, supportando virtualmente tutte le caratteristiche della chiamata convenzionale come: identificativo del chiamante (Caller*ID), chiamata in attesa, identificativo del chiamante su chiamata in attesa, funzione libero/occupato, libero/senza risposta, call forward variable, stutter dialtone, chiamata a tre, trasferimento supervisionato e non, supporto ADSI (Analog Display Services Interface), mail vocali, conferencing, VoIP gatewaying, database con dettagli di chiamata, ecc.

Allo stesso tempo, è un IVR (interactive voice response), cioè offre un servizio di risposta automatico con operatore virtuale, completamente programmabile, come vedremo in seguito.

Le applicazioni IVR collegano insieme un'interfaccia all'altra e non è necessario conoscere nulla circa l'interfaccia fisica, il protocollo o il codec che riguardano la chiamata in corso, dal momento che Asterisk fornisce una totale astrazione di tutti questi concetti, comportandosi, dunque, come una black box.

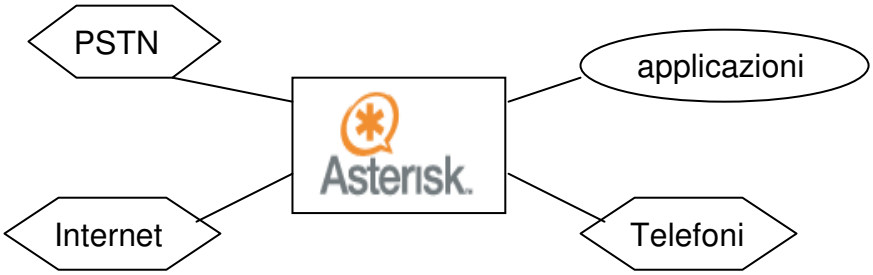


Fig. 11 – Asterisk visto come una 'black box'

» 2.2 Funzionalità di Asterisk

Asterisk ha tutte le funzionalità di un PBX tradizionale e di un gateway, a cui se ne aggiungono altre tipiche di sistemi telefonici avanzati:

- Segreteria telefonica (con integrazione dei servizi di posta elettronica);
- Funzioni giorno/notte/festivo/pausa completamente personalizzabili e flessibili;
- Risponditore telefonico multilivello completamente programmabile (IVR);
- Caselle vocali personalizzate;
- Annunci vocali personalizzati;
- Supporto del CallerID (identificativo del chiamante) anche sulle chiamate in attesa;
- Gestione delle chiamate in attesa;
- Gestione di servizi di call-back;
- Gestione LCR: Least Cost Routine (instradamento delle chiamate verso l'operatore più economico per orario e tipo di chiamate, incluso instradamento verso schede GSM);
- Funzionalità ACD: Automated Call Distribution (permette alle chiamate in ingresso di essere gestite in modo equo);
- Conversazione a tre;
- Possibilità di gestione fax server (con inoltro automatico via email);
- Funzioni di teleconferenza;
- Interfacciamento con i software aziendali per applicazioni complesse (come ad esempio il riconoscimento e la visualizzazione della scheda cliente in base al numero del chiamante o accesso a qualsiasi tipo di contenuto dinamico);
- Funzionalità complete VoIP;

- Servizi di autenticazione (accesso a servizi tramite password);
- Funzioni di telemanutenzione opzionali per ridurre al minimo la necessità di interventi in sede;
- Possibilità di sviluppo di nuove funzionalità con minimo sforzo grazie alla piattaforma aperta su cui si basa il sistema;
- Possibilità di abbinare, sulla stessa macchina e senza decadimento delle prestazioni, un firewall per la protezione della rete dati dell'azienda.

» 2.3 Architettura di Asterisk

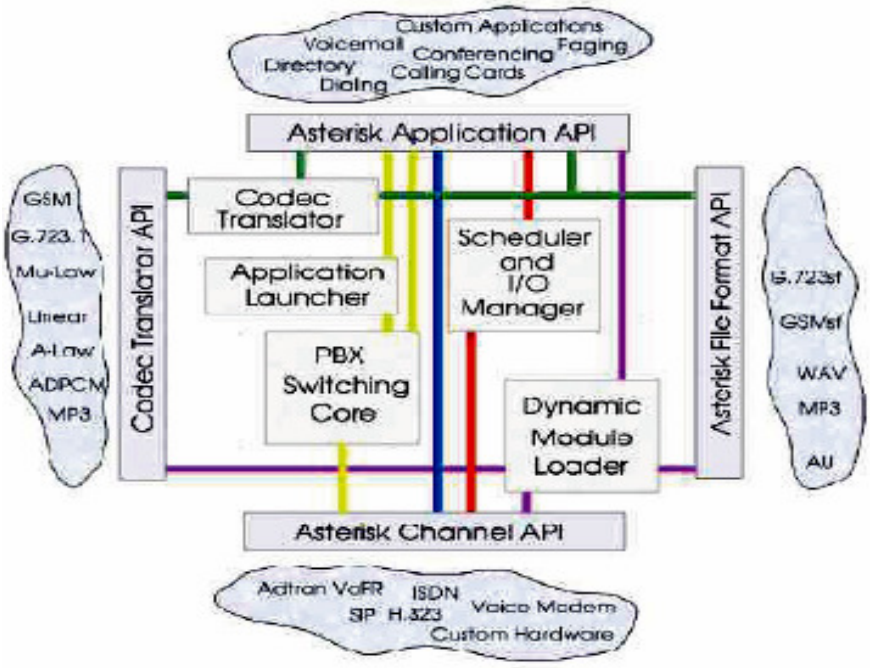


Fig. 12– Architettura interna di Asterisk

Asterisk è realizzato per garantire il massimo della flessibilità. API (Application Program Interface – set di routines, protocolli e tools per creare applicazioni software) specifiche sono definite attorno ad un

nucleo PBX centrale, il quale si occupa della gestione delle interconnessioni interne del PBX: il tutto avviene in modo astratto dai protocolli specifici, dai codec e dalle interfacce hardware delle applicazioni telefoniche.

Questo permette ad Asterisk di usare qualsiasi suite di hardware e di tecnologie disponibili ora o nel futuro per fornire le sue funzioni essenziali, connettendo hardware e applicazioni.

Il nucleo di Asterisk (vedi fig. 12) consta dei seguenti blocchi:

- PBX Switching: l'essenza di Asterisk, naturalmente, è quella di fare da Private Branch Exchange Switch, connettendo tra loro chiamate tra vari utenti. Lo Switching Core (nucleo) connette in modo trasparente gli utenti che chiamano utilizzando varie interfacce hardware e software;
- Application Launcher: lancia applicazioni che offrono servizi per utilizzi come la voicemail, il file playback e l'elenco di directory;
- Codec Translator: usa codec caricati come moduli per la codifica e decodifica dei vari formati di compressione audio. Sono disponibili molti codec per soddisfare diversi bisogni e per raggiungere il miglior compromesso tra qualità audio e uso di banda;
- Scheduler and I/O Manager: cura i compiti di basso livello di scheduling (interruzione e avvio di più processi) e di gestione del sistema per garantire performance ottimali sotto ogni condizione di carico.

Continuando ad osservare la fig. 12 esaminiamo i **moduli API caricabili**: quattro API sono state definite per i moduli caricabili, per facilitare l'astrazione dell'hardware e dei protocolli.

Utilizzando questo sistema di moduli caricabili, il nucleo di Asterisk non deve preoccuparsi dei dettagli di come un utente si connette, che codec usa, etc.

I moduli API sono:

- Channel API: cura il tipo di connessione sulla quale arriva una chiamata, che può essere un collegamento VoIP, ISDN, PRI, a segnalazione Robbed bit, o con qualche altra tecnologia. I moduli dinamici sono caricati per curare i dettagli dei livelli bassi di queste connessioni.
- Application API: l'API dell'applicazione permette a vari moduli specializzati di essere inseriti per fornire varie funzioni. Conferenze, Paging, elenco di directory, Voicemail, trasmissione dati in linea, e molte altre funzionalità che un PBX deve offrire ora o nel futuro sono svolte da questi moduli indipendenti.
- Codec Translator API: carica i codec in moduli per supportare vari formati di audio encoding e decoding come il GSM, il compressore a legge μ (usato negli USA) o quello a legge A (usato in Europa), e perfino l'MP3.
- File Format API: si preoccupa della lettura e della scrittura di vari formati di file per la memorizzazione dei dati nel filesystem.

Usando queste API Asterisk raggiunge una completa astrazione tra le funzioni del suo nucleo come server PBX e le varie tecnologie esistenti (o in via di sviluppo) nel mondo della telefonia. La forma modulare permette ad Asterisk di integrare in modo flessibile sia la telefonia tradizionale corrente che le emergenti tecnologie di Packet Voice. L'abilità di caricare i codec in moduli permette ad Asterisk di supportare sia codifiche estremamente compatte necessarie per la Packet Voice nel caso di connessioni lente come con un telefono

modem che codifiche ad alta qualità nel caso di connessioni meno limitate.

Le application API offrono un uso flessibile dei moduli delle applicazioni per fornire ogni tipo di funzione “on demand”, e permette uno sviluppo open di nuove applicazioni per soddisfare richieste particolari.

Inoltre, caricare tutte le applicazioni come moduli ne fa un sistema flessibile, permettendo all'amministratore di scegliere i migliori percorsi per le chiamate nel sistema PBX e di modificarli per soddisfare al meglio l'andamento del traffico.

Vediamo ora come interagiscono tra loro i diversi moduli nelle operazioni.

All’avvio, il **Dinamic Module Loader** carica e inizializza i driver per ognuno dei canali, il formato dei file supportati, i record con i dettagli di chiamata di ogni back – end, codec, applicazioni e si collega con le API interne appropriate.

In seguito il **PBX Switching Core** inizia ad accettare le chiamate dalle interfacce e le tratta in accordo con il dialplan, utilizzando l’**Application Launcher** per chiamare i telefoni, connettersi alla voicemail, effettuare chiamate esterne ecc. Il nucleo ha anche uno **Scheduler and I/O Manager** standard a disposizione delle applicazioni e dei driver.

Il **Codec Translator** permette a canali che utilizzano codec diversi di parlare tra loro.

La maggiore utilità e flessibilità di Asterisk derivano dalle applicazioni, dai codec e dai channel driver di cui sono dotate le varie interfacce di programmazione.

» 2.4 Il dialplan di Asterisk

Capire Asterisk significa innanzitutto capire il suo dialplan, l'entità che instrada ogni chiamata nel sistema, dalla sorgente al destinatario finale, passando per le varie applicazioni.

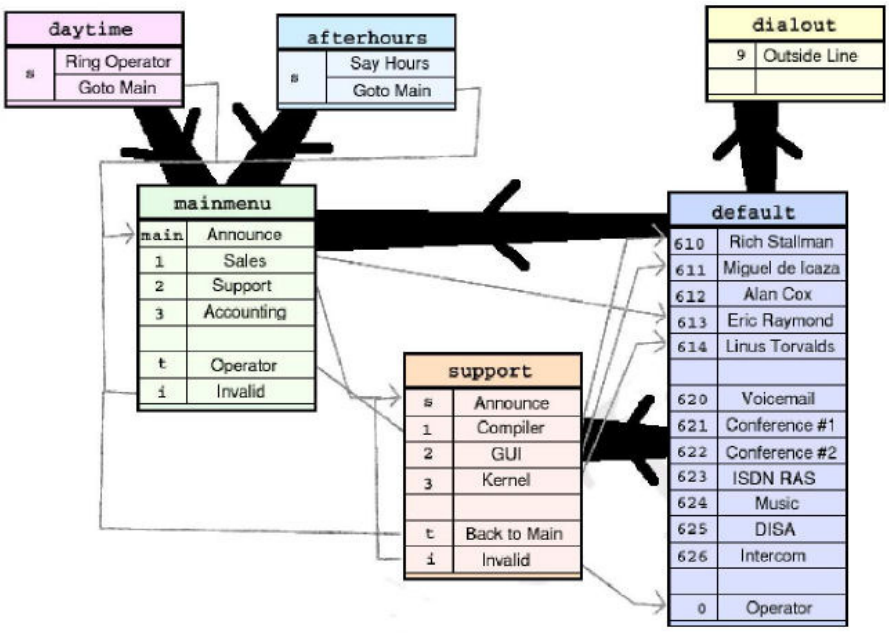


Fig. 13 – esempio di dialplan

Il diagramma in fig. 13 rappresenta un esempio di dialplan completo e conferisce al centralino Asterisk molta flessibilità.

Il dialplan è l'oggetto che determina l'instradamento delle telefonate; esso è composto da un gruppo (generalmente connesso) di contesti, che sono semplicemente collezione di estensioni, cioè istruzioni. Una data estensione può apparire in più contesti e un contesto può includerne un altro.

I contesti possono anche fare riferimento a switch esterni (come ad esempio lo IAX switch) permettendo l'allargamento del sistema verso l'esterno.

Nella figura le linee nere spesse indicano che un contesto è incluso in un altro mentre le sottili linee grigie indicano che un'estensione chiama in un'altra estensione o in un contesto.

In generale, le estensioni che sono locali alla compagnia potrebbero essere comprese sotto il contesto 'default' affinché siano accessibili sempre a chiunque.

Un contesto locale (in questo caso 'dialout') può includere anche quello di default e non solo il 'local' o il 'long distance'.

E' importante che l'accesso al contesto 'local' sia negato ai 'guest' account, a meno che non ci sia esplicita intenzione di permettere l'utilizzo delle risorse telefoniche locali. Successivamente abbiamo una coppia di contesti a menù: 'mainmenù' e 'support'.

Entrambi includono il contesto 'default', in modo da poter chiamare allo stesso tempo le estensioni dirette.

Il contesto 'mainmenù' include sia 'daytime' che afterhours in modo che, se arriva una chiamata durante il giorno squilla il telefono dell'operatore, altrimenti dà un annuncio circa l'orario d'ufficio.

I servizi implementati dai contesti sono vari, e includono importanti funzionalità, come la sicurezza, l'instradamento delle chiamate a seconda dell'estensione, esecuzione di messaggi di istruzioni e benvenuto, autenticazione con password, impostazione di blacklist di utenti che non possono contattare altri, modalità giorno/notte (variazione del comportamento a seconda dell'ora del giorno), registrazione ed esecuzione di macro, ecc.

Il dialplan è specificato nel file di configurazione chiamato extensions.conf, che discuteremo in seguito.

Cominciamo ad analizzare gli elementi costitutivi del dialplan: contesti, estensioni, priorità e applicazioni.

» 2.4.1 Contesti, estensioni, priorità, applicazioni

I contesti, come già detto, sono gruppi di estensioni che suddividono il dialplan in varie parti che possono interagire tra di loro. A meno che sia espressamente previsto, un'estensione definita in un contesto è completamente isolata dalle estensioni di un altro contesto.

Per definire un contesto occorre mettere il nome tra parentesi quadre ([ ]).

Tutte le istruzioni inserite dopo la definizione di un contesto fanno parte di esso, fino a che si giunge alla definizione del contesto successivo.

Uno dei casi più importanti di uso di contesti è la sicurezza: è infatti possibile, configurando i contesti correttamente, limitare ad alcuni utenti alcuni servizi, rendendoli inaccessibili ad altri ed evitando così ad esempio intrusioni non volute al sistema.

Le estensioni sono istruzioni definite all'interno dei contesti, e specificano cosa succede alle chiamate nel loro percorso attraverso il dialplan.

La sintassi utilizzata per le estensioni è:

```
exten =>
```

seguita di regola da tre componenti:

- Il nome dell'estensione;
- La priorità (ogni estensione può includere vari 'step'; il numero di step è appunto la priorità);
- L'applicazione, che mette in atto qualche azione sulla chiamata.

Un possibile esempio è:

```
exten => 123,1,Answer( )
```

dove il nome dell'estensione è 123, la priorità è 1 e l'applicazione è Answer ().

I nomi per alcune estensioni sono riservati a scopi speciali.

Ecco qualche esempio:

- **s**: è l'estensione 'start'. Una chiamata a cui non sono associate delle cifre inizia con 's';
- **t**: è l'estensione 'timeout'. Viene eseguita quando un utente è in menù voce e non digita i tasti opportuni (o non digita nulla);
- **i**: è l'estensione 'invalid'. Viene eseguita quando un utente digita un numero non valido;
- **o**: è l'estensione 'operator'. Viene eseguita quando un utente toglie la segreteria per parlare con un operatore;
- **h**: è l'estensione 'hangup'. Viene eseguita alla fine della chiamata quando uno degli utenti riaggancia.

Le priorità indicano l'importanza e l'ordine nell'esecuzione delle istruzioni. Sono numerate in maniera progressiva, di regola partendo da 1 (con l'eccezione delle priorità 'non numerate', che si indicano con 'n' , che corrispondono alla priorità dell'istruzione precedente + 1, e che evitano di rinumerare l'intero dialplan nel caso di modifiche).

Le applicazioni specificano l'azione da eseguire sul canale corrente, come suonare la musica d'attesa, accettare una chiamata in arrivo, riagganciare. Alcune applicazioni richiedono delle informazioni aggiuntive, gli 'argomenti', specificati tra parentesi dopo il nome dell'applicazione, e separati da una virgola.

» 2.4.2 Introduzione ai contesti fondamentali

Un esempio di contesto, chiamato “default”, è quello mostrato in fig. 14.

In questo caso, i primi tre interni da 101 a 103, sono associati a telefoni appartenenti ai vari principali.

Il quarto, invece, permette di controllare la propria voicemail, il quinto è associato ad una stanza per conferenze e infine lo ‘0’ all’operatore.

default	
Extension	*Description*
101	Mark Spencer
102	Wil Meadows
103	Greg Vance
104	Check voicemail
105	Conference Room
0	Operator

Fig. 14 – contesto default

Vediamo un altro esempio:

mainmenu	
Extension	*Description*
s	Welcome message and instructions
1	Sales
2	Support
3	Accounting
9	Directory
#	Hangup

Fig.15 – contesto mainmenù

Questo contesto chiamato “mainmenù” ha un'unica estensione.

Infatti digitando 's' (start) inizia la chiamata e il sistema risponderà con un messaggio di benvenuto, contenente l'istruzione del tipo "premere 1 per le vendite, 2 per il supporto, 3 per l'accounting, 9 per la compagnia, oppure # per terminare". Dunque ogni opzione del menù è un'estensione che può chiamare altre estensioni sia nello stesso menù, sia in altri menù.

Con le estensioni si possono fare anche confronti tra pattern diversi e non solo tra singole cifre: in questo caso i pattern (cioè i percorsi, le serie alfanumeriche) da confrontare devono iniziare per underscore ("_") e bisogna usare questi caratteri:

- **X**, indica un numero qualsiasi;
- **Z**, indica un numero tra 1 e 9;
- **N**, indica un numero tra 2 e 9;
- **[14-6]**, indica un numero che sia 1, 4, 5 o 6;
- **.**, ovvero il punto, a indicare qualsiasi tasto.

Ecco un esempio di contesto che divide le telefonate in base all'estensione: tutto ciò che inizia con 61 viene instradato all'ufficio di Dallas, con 62 a quello di Huntsville, ecc.

routing	
Extension	*Description*
_61XX	Dallas Office
_62XX	Huntsville Office
_63XX	Dallas Office
_7[1-3]XX	San Jose Office
_7[04-9]XX	Los Angeles Office

Fig.16 – contesto routine

» 2.4.3 Inclusione di contesti

Un contesto può includere altri contesti: un esempio è illustrato nella fig. 17

longdistance	
Extension	*Description*
_91NXXNXXXXXX	Long distance calls
include => "local"	

local	
Extension	*Description*
_9NXXXXXX	Local calls
include => "default"	

Fig.17 – inclusione di contesti

Qui un contesto, "local", definisce una singola estensione per comporre le chiamate locali ed include anche l'estensione "default".

Il contesto "longdistance" include invece un'estensione per le chiamate a lunga distanza e il contesto "local", in modo che i telefoni che sono in "longdistance" sono abilitati a fare sia chiamate locali che a lunga distanza (queste ultime, come si può vedere dalla figura, sono caratterizzate da un numero maggiore di cifre).

In questo modo, con l'inclusione di contesti si può controllare l'accesso, abilitare solo alcuni utenti e non tutti ad un determinato tipo di servizi, e dunque tariffarli in maniera separata.

La sintassi utilizzata per l'inclusione di contesti è la seguente:

```
include => context
```

dove ovviamente si sostituirà a 'context' il nome del contesto.

» 2.4.4 Esempi di applicazioni tipiche

Un esempio elementare di dialplan può essere questo:

```
[incoming]
exten => s,1,Answer( )
exten => s,2,Playback(sample)
exten => s,3,Hangup( )
```

Answer( ) esegue le azioni necessarie affinché il canale possa ricevere la chiamata in arrivo; applicazioni come **Playback( )** o **Background( )** invece servono per eseguire un file sonoro (tipicamente in formato .gsm, nel nostro esempio sample.gsm), come una musica d'attesa; Hangup ( ) pone fine alla telefonata.

L'applicazione **Goto ( )**, invece, è usata per spostarsi da una parte all'altra del dialplan, e dunque per inviare la chiamata ad un altro contesto, estensione, e priorità, indicati in quest'ordine tra parentesi come argomento di Goto ( ).

Vediamo un rapido esempio di dialplan in cui si utilizza Goto ( ):

```
[incoming]
exten => s,1,Answer( )
exten => s,2,Background(messaggio_istruzioni)
exten => 1,1,Playback(digits/1)
exten => 1,2,Goto(incoming,s,1)
exten => 2,1,Playback(digits/2)
exten => 2,2,Goto(incoming,s,1)
```

Quando l'utente chiama, ascolterà il messaggio (messaggio_istruzioni.gsm) del tipo: "Per favore inserisca il numero che vuole chiamare". Se preme 1, sentirà un suono associato al tasto 1, e se preme 2, il suono del tasto 2. Una volta fatto questo, l'istruzione Goto restituirà il controllo all'estensione 's'.

Si può poi completare e rendere più robusto il dialplan appena analizzato con l'introduzione di estensioni 'i' e 't', rispettivamente nel

caso di pressione di un tasto non valido (es. il tasto 3 nell'esempio precedente) e di un timeout.

Diamo uno sguardo ora all'applicazione **Dial ( )**, usata per effettuare la chiamata. Dial ( ) supporta fino a quattro argomenti.

Il primo è il canale che si vuole chiamare: ad esempio con l'estensione

```
exten => 123,1,Dial(Zap/1)
```

Asterisk instaurerà un collegamento sul canale Zap 1, se invece si vogliono chiamare più canali contemporaneamente si usa la lettera '&':

```
exten => 123,1,Dial(Zap/1&Zap/2&Zap/3)
```

Il secondo argomento è il timeout espresso in secondi: Asterisk effettua un tentativo di chiamata per tutto il tempo specificato, una volta scaduto passerà all'istruzione con la priorità successiva.

Una particolarità dell'applicazione Dialplan è che con essa è possibile gestire in maniera differente un canale di destinazione occupato da uno in cui non si è avuta risposta: se il canale è occupato, Dial ( ) punterà all'istruzione con priorità n+101, anziché n+1.

Vediamo un esempio:

```
exten => 123,1,Dial(Zap/1,10)
exten => 123,2,Playback(vm-nobodyavail)
exten => 123,3,Hangup( )
exten => 123,102,Playback(tt-allbusy)
exten => 123,103,Hangup( )
```

Asterisk chiama il canale Zap 1 per 10 secondi, trascorsi i quali, se non si ha risposta, verrà eseguito il file sonoro vm-nobodyavail.gsm. Se il canale è occupato si avrà l'esecuzione del file tt-allbusy.gsm.

Il terzo argomento di Dial ( ) è una stringa opzionale, con uno o più caratteri che modificano il comportamento di Dial ( ) stesso.

Ad esempio, con la lettera 'r' (ringing), si forzerà Asterisk ad eseguire il tono di chiamata, lo squillo, insomma.

Il quarto e ultimo argomento di Dial ( ), raramente usato, è un URL che può essere inviato al canale di destinazione, sempre che quest'ultimo ne supporti la ricezione. Esempio:

```
exten => 124,1,Dial(IAX2/pippo@sito.com)
```

Vediamo ora qualche esempio di estensione logica:

➔ **Estensione con "anti – ex – girlfriend"**

```
exten => 100/2565551212,1,Congestion
exten => 100,1,Dial(Zap/1,20)
exten => 100,2,Voicemail(u100)
exten => 100,102,Voicemail(b100)
```

Questo è un esempio di instradamento delle chiamate a seconda dell'ID del chiamante. Se l'ID dell'utente chiamante è 256-555-1212, allora gli verrà subito riprodotto un 'tono di congestione', come se il numero fosse non valido o avesse un problema. Altrimenti, l'applicazione Dial è utilizzata per chiamare Zap/1 per 20 secondi, scaduti i quali si va alla Voicemail (applicazione che permette di eseguire messaggi vocali) 'occupato' ('b' sta per busy, occupato) oppure senza risposta ('u' sta per unavailable, non disponibile).

➔ **Linea in uscita con routine al minimo costo (least cost routing)**

```
exten => _9NXXXXXX,1,Dial(Zap,g2,BYEXTENSION)
exten => _9NXXXXXX,2,Dial(IAX,oh,BYEXTENSION)
exten => _9NXXXXXX,3,Congestion
```

Questo esempio mostra l'uso del confronto tra pattern ed evidenzia come ogni cosa in Asterisk è considerato come un'estensione, sia esso una linea in uscita, un'interfaccia in ingresso o un menù voce. Qui cerchiamo di utilizzare qualsiasi interfacciai nel 'group 2' prima di connettere la chiamata. Se non è disponibile, proviamo a chiamare un altro IAX host chiamato 'oh'. Se anche questo fallisce viene eseguito il tono di congestione.

➔ Integrazione AGI per il routing

```
exten => s,1,AGI,agi-lookup.agi
exten => s,2,Background(intro)


exten => 100,1,AGI,agi-save.agi
exten => 100,2,Dial(Zap/9,15)
exten => 100,3,Voicemail(u100)
. . .
exten => 120,1,AGI,agi-save.agi
exten => 120,2,Dial(Zap/24,15)
exten => 120,3,Voicemail(u101)
```

Questo set di estensioni permette all'utente di connettersi con l'ultima persona con cui ha parlato, senza doversi ricordare del suo numero. Per prima cosa un utente che chiama è indirizzato allo script "agi-lookup", che controlla il suo ID in un database. Se questo utente ha già parlato con qualcuno precedentemente, lo script "agi-lookup" lo indirizza all'ultima estensione che ha chiamato, altrimenti la chiamata procede normalmente. Quando l'utente si connette a qualche estensione, lo script "agi-save" salva la persona con cui si parla per avere un riferimento in futuro nello script "agi-lookup" la prossima volta che viene chiamato.

» 2.4.5 Variabili

Per ridurre errori nella compilazione del dialplan e per aggiungere chiarezza spesso si usano variabili.

Per esempio, se John utilizza il canale Zap/1, allora possiamo creare la variabile John e assegnarle il valore del canale utilizzato, con questa sintassi:

```
john=Zap/1
```

Se ci si vuole riferire al nome di variabile, semplicemente si scriverà il nome della variabile così com'è; se ci si vuole riferire al suo valore, occorre inserire il nome della variabile tra parentesi graffe ({) e mettere davanti il simbolo del dollaro ($). Ecco un esempio:

```
exten => 555,1,Dial(${JOHN},,r)
```

Esistono tre tipi di variabile:

- **variabili globali**: si possono applicare a qualsiasi estensione e in qualsiasi contesto. Sono molto utili laddove occorra sostituire in un dialplan centinaia di riferimenti ad uno stesso oggetto in modo semplice, veloce, senza rischio di errori. Possono essere definite sia in un contesto a loro riservato, [global], all'inizio del file di configurazione extensions.conf, oppure usando l'applicazione SetGlobalVar;
- **variabili di canale**: sono associate e disponibili solo per una particolare chiamata, e per il canale utilizzato nella chiamata stessa. Un esempio è l'ID dell'utente;
- **variabili di sistema**: permettono di riferirsi ad una variabile del sistema Unix; la sintassi utilizzata è `${ENV(var)}`, dove 'var' è la variabile Unix a cui ci si vuole riferire.

» 2.5 Organizzazione dei file

L'organizzazione di Asterisk segue quella classica di un sistema Linux, ed è raggruppata in diverse directory:

- /**etc**/**asterisk** : questa directory contiene tutti i file di configurazione di Asterisk, come extension.conf, sip.conf, oh323.conf e iax.conf;
- /**usr**/**sbin** : la directory dei file binari di sistema contiene gli eseguibili e gli script effettivi, tra cui asterisk, astman, astgeney e safe_asterisk;
- /**usr**/**lib**/**asterisk** : contiene oggetti binari riferiti ad Asterisk specifici della sua architettura;
- /**usr**/**lib**/**asterisk**/**modules** : contiene i moduli runtime per applicazioni, channel driver, codec, driver per il formato dei file, ecc;
- /**usr**/**include**/**asterisk** : contiene gli header dei file richiesti per la costruzione delle applicazioni, channel driver e altri moduli caricabili;
- /**asterisk**/**doc**/**agi**/**html**/**index.html** : contiene la documentazione della API interne di Asterisk. E' un file di indice navigabile con un browser.
- /**var**/**lib**/**asterisk** : contiene le variabili utilizzate da Asterisk nelle sue normali operazioni.
- /**var**/**lib**/**asterisk**/**agi-bin** : contiene gli AGI script usati nelle applicazioni AGI del dialplan.
- /**var**/**lib**/**asterisk**/**astdb** : è il database di Asterisk, più o meno equivalente al registro di Windows. Non è mai utilizzato direttamente, ma il suo contenuto può essere visualizzato e modificato dalla linea di comando con il set di funzioni "database";
- /**var**/**lib**/**asterisk**/**images** : area per la memorizzazione di immagini riferite al dialplan e alle applicazioni;

- /**var**/**lib**/**asterisk**/**keys** : area per la memorizzazione di chiavi pubbliche e private utilizzate per l'autenticazione RSA con Asterisk (specialmente IAX);
- /**var**/**lib**/**asterisk**/**mohmp3** : area per la memorizzazione di mp3 musicali. Qui sono contenuti tutti gli mp3 che si vuole abilitare a riprodurre.
- /**var**/**lib**/**asterisk**/**sounds** : area per la memorizzazione di audio file, suggerimenti, ecc. utilizzati nelle applicazioni di Asterisk. Alcuni suggerimenti inoltre sono organizzati in sottodirectory /var/lib/asterisk/sounds.
- /**var**/**lib**/**sounds**/**voicemail** : contiene tutti i messaggi vocali relativi ad ogni utente e anche quelli di testo che visualizzano i dettagli del messaggio;
- /**var**/**run** : directory di sistema contenente i process ID (PID) per tutti i processi attivi. E' dipendente dal sistema operativo utilizzato;
- /**var**/**spool**/**asterisk** : utilizzato per file utilizzati a runtime come voicemail, chiamate in uscita, ecc.
- /**var**/**spool**/**asterisk**/**outgoing** : controllato da Asterisk per chiamate in uscita. Quando viene creato un file in questa directory, Asterisk suddivide il file in ingresso e prova ad effettuare una chiamata in uscita che viene indirizzata nel PBX se è richiesto. Ha sostituito il vecchio sistema basato sulla directory /var/spool/asterisk/qcall;
- /**var**/**spool**/**asterisk**/**vm** : memorizza caselle di voicemail, annunci e cartelle.

» 2.6 File di configurazione

Asterisk, come abbiamo già detto, viene configurato mediante un certo numero di file situati in /etc/asterisk.

Questi hanno una sintassi formata da una o più sezioni (il cui titolo è racchiuso tra parentesi quadre ([ ]) e da variabili e oggetti (che si dichiarano con "=" e con "=>"); i commenti sono preceduti da punto e virgola.

» 2.6.1 Classificazione

Possiamo classificare i file di configurazione di Asterisk in base alla sintassi:

- **Simple group**. E' il formato più semplice, con oggetti dichiarati con tutte le opzioni su un'unica riga. Esempi di file appartenenti a tale categoria sono extensions.conf, meetme.conf, voicemail.conf, ecc.

  Per esempio:

  ```
  [mysection]
  object1 => option1a,option2a,option3a
  object2 => option1b,option2b,option3b
  ```

  Dunque l'oggetto object1 è dichiarato con le opzioni option1a, 2a, 3a, mentre l'oggetto object2 con le opzioni option1b, 2b, 3b.

  L' "Individual Entities" è un tipo di sintassi usata nei file di configurazione in cui ogni sezione è associata ad uno specifico oggetto, che avrà poi diverse opzioni, raramente condivise con altri oggetti.

- **Inherited Option Object**. Questo formato (usato ad esempio nei file zapata.conf, phone.conf, mgcp.conf) prevede che la maggior parte degli oggetti condivide le stesse opzioni.

Esistono uno o più sezioni che contengono le dichiarazioni di uno o più oggetti, le cui opzioni sono specificate prima della dichiarazione degli oggetti stessi. E' poi possibile cambiare delle opzioni per degli oggetti che ancora devono essere dichiarati, senza che le modifiche interessino gli oggetti dichiarati precedentemente.

Per esempio:

```
[mysection]
option1 = foo
option2 = bar
object => 1
option1 = baz
object => 2
```

Dopo aver specificato l'opzione 1 e 2 rispettivamente con i valori 'foo' e 'bar', viene dichiarato l'oggetto 1 e settato con tali opzioni. Poi viene cambiata l'opzione 1 (il suo valore da 'bar' diventa 'baz') e dunque dichiarato l'oggetto 2, che avrà l'opzione 1 modificata. Tale modifica non riguarderà l'oggetto 1, perché dichiarato precedentemente.

- **Complex Entity Object**. E' usato in file come iax.conf, sip.conf, e altre interfacce che presentano numerose entità non collegate fra loro, e con ognuna il proprio contesto, con opzioni distinte da altri contesti.

A questo punto, vediamo, come esempio di file di configurazione, i file zapata.conf e extensions.conf.

» 2.6.2 Zapata.conf

Nel file zaptel.conf si definisce il tipo di segnalazione che il canale andrà ad usare, e che canale verrà caricato.

Il file zapata.conf è usato per determinare la configurazione per l'hardware telefonico installato nel sistema, e per controllare parametri come il Caller ID, le chiamate in attesa, cancellazione dell'eco, e altre opzioni.

Quando si configura e si carica zaptel.conf, Asterisk non è in grado ci conoscere cosa si è settato: solo con zapata.conf si potrà informare Asterisk sull'hardware e i relativi parametri che sono stati specificati.

Eccone un esempio:

```
[channels]
usecallerid=yes
hidecallerid=no
callwaiting=no
threewaycalling=yes
transfer=yes
echocancel=yes
echotraining=yes
; define channels
context=incoming
signalling=fxs_ks
callerid = "Mario Rossi" < (256) 555 - 1000 >
channel => 1


callerid = "Franco Verdi" < (256) 555 - 2000 >
callwaiting=yes
channel => 2
```

La sezione [channels] determina i metodi di segnalazione per i canali hardware e le loro opzioni. Una volta che un'opzione è definita o modificata, è ereditata da questo punto in poi, nel resto del file.

In questo esempio abbiamo abilitato il Caller Id e specificato che non deve essere nascosto per le chiamate in uscita; inoltre abbiamo disattivato il callwaiting per il canale 1 (per il telefono di Mario Rossi)

e lo abbiamo attivato per il canale 2 (per il telefono di Franco Verdi), inoltre abbiamo abilitato la chiamata a tre (che di default è disabilitata) e attivato il trasferimento di chiamata.

Altre opzioni presenti nel file riguardano la cancellazione dell'eco (con l'opzione 'echotraining=yes' che velocizza i tempi necessari ad Asterisk a capire il livello di eco nel canale per poi poterlo cancellare in modo efficace.

Per definire delle azioni su un particolare gruppo di chiamate, come quelle in arrivo, si utilizzano i contesti (nel nostro caso utilizzando "context=incoming") che sono specificati nel file extensions.conf.

Infine, dal momento che il canale in questione è FXO (Foreign eXchange Office) e usa una segnalazione FXS (Foreign eXchange Station)*, si utilizza l'espressione "signalling=fxs_ks".

» 2.6.3 Extensions.conf

Come già detto, nei file che usano la sintassi del 'Simple Group', ogni sezione rappresenta un gruppo che contiene linee singole che definiscono un oggetto che è completamente indipendente dagli altri oggetti nel gruppo.

Un esempio di file è extensions.conf, di cui analizziamo qualche riga come modello:

```
     [context1]
;exten => estensione, priorità, applicazione;
exten => s,1,Wait,3
exten => s,2,Answer
exten => s,3,Voicemail,u600
exten => 100,1,Dial,Zap/g2
```

* Per spiegazioni su porte FXS e porte FXO vedere il paragrafo 2.10.1

```
     [context2]
exten => 9,1,Dial,Zap/g2/9
include => context1
include => context3

     [context3]
switch => IAX/myuser@ohterhost/local
```

In questo esempio vengono dichiarati tre contesti. Il context1 ha due estensioni, ‘s’ con tre step, e ‘100’ che ne ha uno.

Il context2 include l’estensione ‘9’ e qualsiasi cosa dei context1 e del context3 (notare che le inclusioni vengono esaminate in ordine come sono elencate, così se un certo passo di una estensione è presente sia in context1 che in context2, viene eseguito solo il context1).

» 2.6.4 IAX.conf

Questo file contiene tutte le informazioni di cui Asterisk ha bisogno per gestire le connessioni attraverso il protocollo IAX, usato dai server Asterisk per comunicare fra loro.

Iax.conf inizia con una sezione “general”, con i parametri globali come l’indirizzo, il buffer anti-jitter, la porta, poi prosegue con un certo numero di entità ognuna delle quali ha un “type” (user, peer, friend) con i suoi parametri. Ogni entità è completamente indipendente dalle altre entità che la circondano.

```
[general]
bandwidth=low
disallow=lpc10
jitterbuffer=no
forcejitterbuffer=no
tos=lowdelay
autokill=yes
register => fwd_number:password@iax2.fwdnet.net
```

```
[iaxfwd]
type=user
context=incoming
auth=rsa
inkeys=freeworlddialup
```

L'espressione "register" è usata per registrarsi ad un server remoto, permettendo al terminale remoto di sapere il nostro indirizzo IP.

La sintassi utilizzata per questa espressione è:

```
register => username:password@host remoto
```

Altri esempi tipici di configurazioni possibili in 'general' sono "allow" o "disallow" che specificano i particolari codec che si vogliono utilizzare o meno; l'utilizzo del buffer di dejitter; il Tos (type of service), cioè il tipo di servizio realizzato (nel nostro esempio orientato a minimizzare il ritardo); specificare con "autokill" se in caso di irraggiungibilità di un host le connessioni devono essere o no interrotte (il tempo limite di default è di 2 secondi, ma può essere personalizzato).

Nella sezione successiva, [iaxfwd], definiamo l'utente per le chiamate in arrivo con l'espressione "type=user", e definiamo in quale contesto del dialplan le chiamate stesse verranno trattate, con "context=incoming"; per specificare uno dei tipi possibili di autenticazione è stata utilizzata l'espressione "auth=rsa", con la chiave pubblica "freeworlddialup".

Mentre con "type=user" è possibile specificare i parametri per ricevere le chiamate, con "type=peer" si delineano le configurazioni per effettuarle.

Un ulteriore esempio di sezione [iaxfwd] è:

```
[iaxfwd]
type=peer
host=iax2.fwdnet.net
username=<fwd-account-number>
secret=<fwd-account-password>
qualify=yes
disallow=all
allow=ulaw
allow=gsm
allow=ilbc
allow=g726
```

Qui abbiamo definito un entità pari (peer) e usato "host" per configurare il server attraverso il quale poter effettuare le telefonate; abbiamo indicato poi il numero e la password dell'account, necessari per l'autenticazione.

» 2.7 Programmabilità

Ciò che si è visto finora è solo un modo per programmare Asterisk.

In generale ci sono tre livelli di programmazione che, dal livello più alto al più basso, sono:

- **Extension logic**: si modifica il dialplan contenuto in extensions.conf per creare semplici applicazioni, autorizzazioni, ecc.;
- **Asterisk Gateway Interface (AGI)**: per compiti più complessi e sofisticati, AGI permette di lanciare programmi esterni scritti in un qualsiasi linguaggio (es. Perl, Php, ecc.).

  Un Agi script è lanciato dal dialplan. Esso riceve istruzioni dallo stdin che può causare l'emissione di comandi dallo stout e leggere i risultati dallo stdin. In questo caso, non sono richiesti vincoli speciali e qualsiasi linguaggio può scrivere nello stdout e leggere dallo stdin.

- **C – Level API**: Asterisk prevede una API apposita per poter scrivere nel linguaggio C applicazioni particolarmente complesse, channel driver, formati dei file, ecc.

» 2.8 LDAP e Asterisk

L'IETF ha progettato il protocollo LDAP (Lightweight Directory Access Protocol), funzionante su TCP, come un metodo di accesso ai servizi di directory X.500, semplificato rispetto all'originale Directory Access Protocol (DAP), incluso in X.500, troppo complesso per l'uso da parte di semplici applicazioni client internet.

X.500 in sintesi è una serie di directory, cioè elenchi standard su computer per accedere ad elenchi di persone e risorse nel mondo, ottimizzati per l'operazione.

Il protocollo ha influenzato lo sviluppo di alcuni protocolli di rete successivi, come le versioni aggiornate di X.500, il Directory Services Markup Language (DSML), il Service Provisioning Markup Language (SPML) e il Service Location Protocol.

Asterisk prevede diverse integrazione con LDAP.

Asterisk::LDAP è un modulo Perl scritto in un linguaggio orientato agli oggetti per la realizzazione di files di configurazione Asterisk come:

- extensions.conf
- voicemail.conf
- musiconhold.conf

E' possibile inoltre accedere a directory LDAP (come OpenLDAP o ActiveDirectory) dal Dialplan attraverso l'applicazione LDAPget (), non inclusa di default in Asterisk, ma occorre dal sito *http://www.mezzo.net/asterisk/app_ldap.html* scaricare e installare le relative librerie e i file header.

La sintassi per LDAPget nel file extensions.conf è:

```
LDAPget(VARIABLE=config-file-section[/lookup-
key1,lookup-key2,...])
```

Dunque per esempio si potrà aggiungere:

```
exten => 1234,1,LDAPget(CALLERIDNAME=cidname)
exten => 1234,2,Dial(Zap/11/${EXTEN},15)
```

Poi occorre configurare anche la corrispondente sezione in ldap.conf file, dove aggiungeremo il contesto [cidname].

```
[cidname]
host = www.dominio.com; (default: localhost)
port = 389; (default: 389)
timeout = 10; (default: 10 sec)
version = 2; (default: 2)
user = cn=root,ou=People,o=www.dominio.com
pass = secret
base = ou=Addressbook,o=www.domain.com
filter =
(&(objectClass=person)(|(telephoneNumber=${CALLERIDNUM}
)(mobile=${CALLERIDNUM})(homePhone=${CALLERIDNUM})(fax=
${CALLERIDNUM})))
convert = UTF-8,ISO-8859-1
```

LDAP si connetterà a www.dominio.com attraverso la porta di default 389 (entrambi questi parametri, dominio e porta, sono comunque modificabili) con i nomi utente e password indicati (se si omettono LDAP si connetterà in modo anonimo).

A questo punto LDAP eseguirà una ricerca, al termine della quale LDAP modificherà la variabile CIDNAME.

In aggiunta a ciò va ricordata l'applicazione Authenticate_LDAP, che verifica la disponibilità di un parametro in un database LDAP.

» 2.9 Esempi di network

A questo punto è utile esaminare alcune installazioni reali di Asterisk, evidenziando come la scalabilità permetta di passare da un'architettura piccola ad una molto grande in modo semplice.

» 2.9.1 PBX 1x1

Fig. 18 – PBX 1x1

La figura 18 mostra come una scheda X100P della LSS, o una Linejack della Quicknet può essere utilizzata per portare una linea telefonica in ingresso ad un PC, su cui è in esecuzione Asterisk, che è collegato ad un telefono analogico e ad un telefono IP. Come un PBX classico, Asterisk offre i servizi di voicemail, si comporta da gateway oppure da IVR.

» 2.9.2 Un piccolo ufficio 8x16

Fig. 19– 8x16 small office PBX

Un più tipico scenario da ufficio è quello con una maggiore densità di telefoni interni rispetto alla rete esterna. In fig. 19 ci sono 8 linee telefoniche e 16 telefoni analogici in ingresso ad un channel bank che multipla i canali in una singola interfaccia T1, collegata ad un PC Linux con Asterisk (con una T1 card come la T100P). Inoltre c'è la possibilità di connettere telefoni IP per aumentare il numero di telefoni impiegati. Persino con un piccolo setup è possibile godere di tutte le potenzialità di Asterisk.

» 2.9.3 Piccole o medie imprese con uffici remoti

Fig. 20 – Piccole o medie imprese con uffici remoti

Con Asterisk c'è la possibilità di collegare uffici remoti di piccole o medie imprese.

La fig. 20 mostra come è possibile costruire un PBX individuale per ogni ufficio per poi collegarli insieme, in modo semplice, in una singola rete.

» 2.9.4 IVR ad alta densità e conferencing

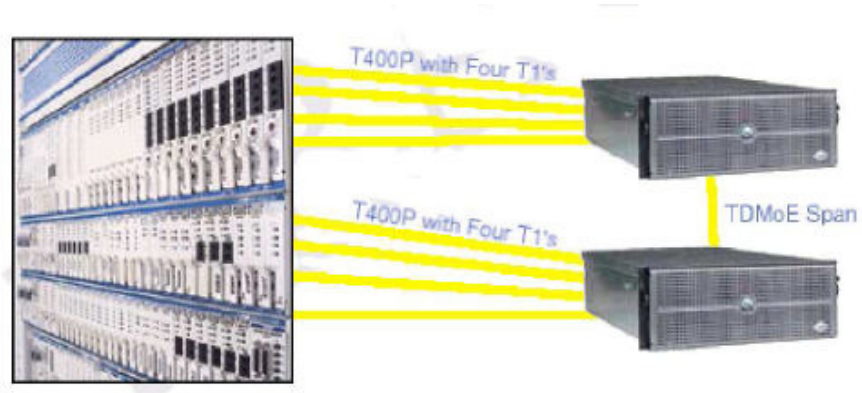


Fig. 21 – IVR ad alta densità e conferencing

Asterisk può essere anche utilizzato per creare un IVR ad alta densità e come piattaforma per conferenze, mediante la tradizionale interfaccia PRI/T1 e fornendo ridondanza, scalabilità, utilizzando TDM over ethernet, che permette ad Asterisk di estendere il bus TDM attraverso la rete, con la minima latenza.

» 2.10 Hardware supportato

Asterisk supporta molte interfacce hardware per inserire canali telefonici in un sistema Linux.

Di seguito, dopo una breve spiegazione sulla differenza tra porte FXS e porte FXO, sono riportati gli hardware più supportati.

» 2.10.1 FXS e FXO

Un dispositivo FXS inizializza e invia un impulso di tensione (il tono di chiamata), ricevuto da un dispositivo FXO.

Il telefono che riceve la chiamata è l'ultimo dispositivo FXO nel canale tra chiamante e chiamato, e nel momento in cui riceve il tono di chiamata da un dispositivo da un dispositivo FXS, il telefono squilla.

Dunque occorre connettere il telefono ad una porta FXS sul server di Asterisk, e quando il modulo FXO presente nel server stesso riceve il tono, genererà tensione usando il modulo FXS e lo invierà al telefono analogico.

Una porta FXO si comporta come una 'stazione,' si può connettere con una linea telefonica (per esempio quella della propria compagnia telefonica) e usa una segnalazione FXS; una porta FXS si comporta come un 'ufficio centrale', si può connettere con un telefono analogico e usa una segnalazione FXO.

» 2.10.2 Le interfacce Zaptel (Zapata Telephony)

L'infrastruttura dei telefoni Zaptel viene sviluppata congiuntamente da Mark Spencer del Linux Support Service Inc. e da Jim Dixon della Zapata Telephony. L'interfaccia Zaptel si differenzia dalla normale filosofia di progettazione perché utilizza un processore dedicato per simulare il bus TDM.

L'architettura pseudo – TDM risultante è vero che richiede qualche processore in più, ma fornisce un risparmio sostanziale nel costo dell'hardware e un incremento della flessibilità. Risorse come cancellatori d'eco, controller HDLC, DSP per conferenza. DAX e molte altre sono sostituite da software equivalenti. La commutazione, come in una tradizionale interfaccia hardware TDM, è effettuata in quasi tempo reale e la qualità della chiamata è sostanzialmente la stessa. La pseudo architettura TDM può anche estendere il bus TDM attraverso una rete Ethernet.

I dispositivi Zaptel supportano PPP, e Frame Relay.

Le interfacce della Digium (tutte Zaptel compatibili) attualmente sono:

» 2.10.2.1 Interfacce digitali

➔ **Digium Wildcard B410P**

E' una scheda S/T a 4 porte, che possono essere configurate separatamente per il modo TE o NT. Tra le funzioni offerte da questa scheda troviamo la cancellazione d'eco.

➔ Digium Wildcard TE110P

La TE110P è un'interfaccia che supporta, mediante apposita selezione, lo standard T1 (24 canali), E1 (32 canali), o J1 (24 canali), sia in modalità voce e sia dati.

Utilizzando la tecnologia TDMoE (TDM over Ethernet) è possibile connettere più PC dotati della TE110P e permettere chiamate voce di qualità tra entità paritarie con una singola implementazione del PBX. Questa interfaccia supporta, tra l'altro, il protocollo PRI (Primary Rate ISDN), il PPP, Cisco HDLC e la modalità Frame Relay.

➔ Digium Wildcard TE205P, TE207P, TE210P, TE212P

Le interfacce TE205P e TE207P supportano sia E1 che l'ambiente T1/J1, mediante una selezione basata su singola interfaccia o su singola porta. Tra i vantaggi nell'uso di queste schede va annoverato un uso più ridotto della CPU, il supporto degli standard telefonici e i protocolli di dati, compreso PRI (sia nello standard americano che europeo), PPP, Cisco, HDLC.

Il TE207P rispetto alla TE205P permette una maggiore qualità del suono, grazie ad un più efficiente metodo di cancellazione d'eco.

La differenza tra la TE210P, la TE212P e le altre interfacce consiste, oltre che nel supporto del modulo VPM450M Octasic DSP-based che permette una efficiente cancellazione d'eco, nello slot supportato, a 3.3 V, presente generalmente nelle schede madri di ultima generazione e nei sistemi a 64 bit.

➔ **Digium Wildcard TE405P, TE407P, TE410P, TE412P**

Praticamente gli analoghi delle interfacce appartenenti alla serie TE2XXP, solo che sono a 4 porte anziché 2.

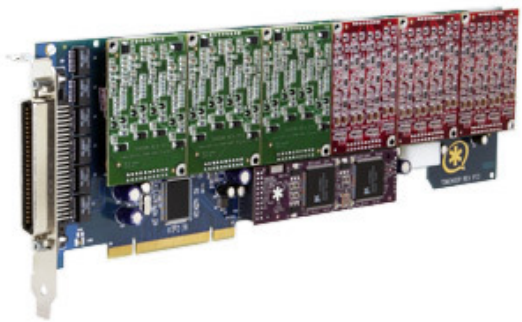
» 2.10.2.2 Interfacce analogiche

➔ **Digium Wildcard TDM400P**

Questa scheda ha 4 porte, e si può comportare, a seconda del modulo usato, o da interfaccia FXO o da FXS, e quindi connettere un telefono analogico o una linea analogica ad un computer.

Richiede l'uso di un connettore da 12 V per le operazioni sui moduli FXS, mentre per quelle sui moduli FXO non è obbligatorio.

➔ **Digium Wildcard TDM2400P**



Anche questa scheda ha 4 porte, e permette il collegamento di telefoni analogici e una linee telefoniche analogiche attraverso un PC. Supporta una combinazione di fino a 6 moduli FXS e/o FXO per un totale di 24 linee, e, se accompagnato da un modulo opzionale, può supportare una cancellazione d'eco migliore rispetto a quello di altre interfacce.

Richiede l'uso di un connettore da 12 V per le operazioni sui moduli FXS, mentre per quelle sui moduli FXO non è obbligatorio.

» 2.10.2.3 Moduli analogici

➔ X100M e S110M

È un modulo FXO che permette il collegamento tra una scheda TDM400P e una linea telefonica analogica (POTS), mentre l'S110M è il corrispondente modulo FXS (e dunque permette il collegamento tra una scheda TDM400P e un telefono analogico).

➔ X400M e S400M

L'X400M è un modulo FXO che permette di collegare una scheda TDM2400P fino a 4 linee telefoniche analogiche per modulo, mentre l'S400M è il suo analogo FXS (e dunque permette di collegare la scheda TDM2400P a telefoni analogici.

» 2.10.3 Interfaccia telefonica per Linux

L'interfaccia telefonica per Linux fu sviluppata inizialmente da Quicknet Inc. con l'aiuto di Alan Cox e viene progettata intorno ad una singola interfaccia analogica, provvedendo anche al supporto per codec a basso bit-rate.

I seguenti prodotti sono compatibili con Asterisk:

- Quicknet Internet Phonejack (ISA, FXS)
- Quicknet Internet Phonejack PCI (PCI, FXS)
- Quicknet Internet Linejack (Isa, FXO o FXS)
- Quicknet Internet Phonecard (PCMCIA, FXS)
- Creative Labs VoIP Blaster (limited support)

➔ ISDN4Linux (ISDN per Linux)

L'interfaccia ISDN4Linux è utilizzata in Europa per portare le linee ISDN dalle interfacce BRI (Basic Rate ISDN) all'interno di Asterisk.

Qualche adattatore supportato da ISDN4Linux potrebbe essere compatibile con Asterisk.

➔ OSS / ALSA Console Driver

Permettono ad una singola sound card di essere utilizzata come una "console telefonica" fare e ricevere chiamate di test. Utilizzando autoanswer/autohangup, la console può anche essere usata per fare da citofono.

➔ Adtran Voice over Frame Relay

Asterisk supporta il protocollo proprietario Atran per il Voice over Frame Relay. E' noto che i prodotti Adtran Atlas 800, 800+ e 550, funzionano bene con Asterisk, a patto di munirsi di un Sagoma Wanpipe o di altre interfacce frame relay.

» 2.11 Tipi di telefoni

Ogni dispositivo fisico con l'obiettivo principale di portare a termine una comunicazione audio on – deman su un circuito che collega due punti, è classificato come telefono.

Vediamo ora una breve classificazione dei vari dispostivi che si possono connettere ad un sistema Asterisk.

» 2.11.1 Telefoni analogici

In questo tipo di apparecchi (gli unici presenti sul mercato fino a 20 – 25 anni fa) il segnale elettrico trasmesso è analogo e proporzionale all'onda sonora prodotta dall'apparato vocale di chi parla.

» 2.11.2 Adattatori

Sono apparecchi hardware che permettono di interfacciare qualsiasi telefono analogico alla rete VoIP convertendo il segnale analogico in formato compatibile. Di solito gli adattatori dispongono di un ingresso di rete per il collegamento alla LAN interna e di un collegamento RJ11 in uscita verso il telefono tradizionale.

E' possibile interfacciare qualsiasi apparato telefonico: cordless, telefono o FAX, consente di convertire in pochi istanti tutti i telefoni di casa in telefoni IP e di effettuare/ricevere chiamate anche col PC spento (nel caso di collegamento a Internet tramite router). Solitamente lo stesso provider VoIP propone l'acquisto o fornisce in comodato l'apparecchio più adatto all'offerta.

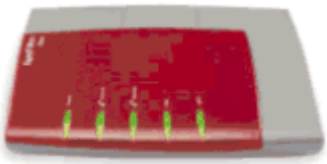
» 2.11.3 Telefoni IP

Sono terminali esteriormente molto simili ai tradizionali apparecchi telefonici ma che collegati ad una rete Lan permettono di effettuare e ricevere telefonate senza la presenza di un computer acceso.

Il mercato offre ormai numerosi modelli, con prezzi variabili a seconda delle funzioni, sono collegabile direttamente a una presa di rete ethernet RJ 45.

» 2.11.4 Dispositivi all-in-one

Questa tipologia di apparecchi integra le funzioni di modem/router Adsl, firewall e adattatore per telefoni analogici, offrendo in alcuni casi la possibilità di collegamento alla linea telefonica tradizionale. Grazie alla possibilità di configurare le regole di instradamento in fase di configurazione, i dispositivi telefonici possono così essere indirizzati sulla linea Pstn o su quella VoIP in base al numero chiamato.



In questo modo sono abilitate le chiamate d'emergenza e verso i numeri speciali (non effettuabili al momento su linee VoIP), ma è anche possibile definire delle regole personalizzate basate ad esempio sui piani tariffari dei provider.

Prodotti di questo tipo sono già presenti sul mercato, o attraverso le opzioni di comodato d'uso dei provider o per l'acquisto diretto da parte dell'utente finale.

» 2.11.5 Telefoni ISDN

Precedenti all'avvento del VoIP, sono stati sviluppati dall'inizio degli anni '80.

Nonostante il largo impiego da parte delle compagnie telefoniche, non in pochi condannano lo standard puntando contro gli elevati costi di implementazione.

In effetti, specie negli ultimi tempi, l'uso di questo tipo di telefonia è in netto declino, in favore di tecnologie come ADSL, e VoIP.

BRI è ancora molto popolare per l'uso in videoconferenza, in quanto provvede ad un collegamento con occupazione di banda fissa, con commutazione di pacchetto.

L'ISDN (Integrated Services Digital Network) supporta due tipi di canali, il canale B e il canale D:

- i canali B (dall'inglese Bearer) sono utilizzati per i dati (che possono essere anche di fonia digitale) ed hanno una banda prefissata di 64 Kbit/s;
- i canali D (da Delta) sono utilizzati per segnalazione e controllo (ma possono essere utilizzati anche per i dati). La banda assegnata al canale D varia a seconda del tipo di accesso (BRI, PRI, T1).

Esistono due tipi di accesso ISDN:

- l'accesso Basic Rate Interface (BRI) - che consiste di due canali B a 64 kbit/s e di un canale D a 16 kbit/s, indicata anche come 2B+D,
- l'accesso Primary Rate Interface (PRI) - che contiene un numero più alto di canali, a seconda del paese:
    - Nord America e Giappone: 24 time-slot da 64kbit/s divisi in 23B+1D, per un bit rate complessivo di 1,544 Mbit/s (T1)
    - Europa, Asia e Australia: 32 time-slot da 64 kbit/s divisi in 30B+1D, bit rate complessivo di 2,048 Mbit/s (E1) - E1 utilizza un time-slot per uso interno, sincronizzazione.

Le chiamate vengono effettuate tramite il canale D e i canali B trasportano i dati quando le due utenze sono connesse. Quando una chiamata viene stabilita si instaura un flusso di dati sincrono e bidirezionale di 64 kbit/s che viene mantenuto fino al termine della chiamata. Possono svolgersi chiamate fino alla disponibilità di altri

canali dati liberi, sempre verso uno stesso punto o verso altre utenze. I canali B possono essere utilizzati in multiplex, realizzando praticamente una singola connessione con banda più elevata.

Il collegamento ISDN più diffuso è quello base a 2 canali (BRI): il doppino telefonico trasporta una banda di 144 Kbit/s suddivisa in due canali dati telefonici da 64 Kbit/s e un canale di segnalazione a 16 Kbit/s: se necessario, i due canali possono essere sfruttati in coppia per trasmissione dati (accesso a Internet, per esempio), pagando di solito tariffa doppia.

Il segnale ISDN non viene fornito direttamente all'apparecchio telefonico come avveniva con la telefonia analogica ma passa attraverso un terminatore di rete (NT, Network Terminator): La NT ha il compito primario di interfacciare la linea proveniente dalla centrale, che viaggia sul classico doppino telefonico chiamato Bus U, all'impianto ISDN interno, che viene realizzato con un cavo a 4 coppie chiamato Bus S, a cui è possibile connettere telefoni e apparati ISDN.

» 2.11.6 Soft Phones

Un softphone è un software che realizza funzionalità telefoniche su di un dispositivo non telefonico, come un PC o un palmare.

Una tipica applicazione dei softphone è quella di effettuare chiamate attraverso un provider ad altri softphone (servizio tipicamente offerto gratis) o a telefoni fissi e cellulari (servizio a pagamento nella maggior parte dei casi).

Altri tipi di telefoni sono collegati a PBX attraverso Lan e sono usati per ricevere ed effettuare telefonate attraverso un esistente hardware telefonico, come in un call-center.

E' importante sottolineare le differenze esistenti tra i softphone e i servizi basati su softphone.

Skype, Google Talk e Vonage sono dei provider telefonici su Internet, ma i loro softphones non permettono di fare chiamare dirette fra di loro (Skype verso Google Talk, per esempio).

Per comunicare fra di loro, i terminali devono avere lo stesso protocollo di comunicazione e almeno un codec audio in comune. Molti provider usano come protocollo SIP, ma altri, come Skype, fanno ricorso a protocolli proprietari, chiusi.

Ecco alcuni tra i più importanti sistemi per il VoIP:

Programma	Sistema operativo	Codice / licensa
3cx_phone_system	Windows XP, 2000, 2003.	Chiuso / SIP
Abbeyphone	Windows XP/2000	Proprietario / SIP
AGEphone	Windows XP/2000, PocketPC, Windows Mobile	Proprietario / Closed
Asterisk PBX	Linux for PPC, OpenBSD, FreeBSD, Mac OS X Jaguar.	Doppia Licensa: GPL / Commerciale
Coccinella	FreeBSD, Linux, Mac OS X, Windows	GPL Free software
Cornfed SIP User Agent	Linux	Gratuita per uso non commerciale
Dingotel	Windows	Codice chiuso
Ekiga	Linux	GPL Free software
Express Talk	Windows	Proprietario freeware
Eyeball Chat	Windows	Proprietario freeware
GameComm	?	
Gizmo	Linux, Mac, Windows XP/2000	Proprietario freeware
GE2006	Windows Only	Proprietario freeware
Google Talk	Windows XP/2000	Il protocollo è open source (libjingle), il resto è Proprietario
iCall	Windows XP/2000	Proprietario freeware
IP Multimedia Subsystem	?	

iVisit	Windows & Mac	Proprietario/closed
Jajah		Proprietario freeware, di recente è stato vietato il download
Kapanga	Windows XP/2000	Gratuita per uso non commerciale
KCall	Linux	GPL / LGPL
KPhone	Linux	GPL
MindSpring (formerly Vling)	Windows XP, 2000,	open SIP, freeware
Minisip	Windows XP, 2000, Linux, Pocket PC	LGPL, GPL
OpenH323 and OPAL	?	Open source/MPL
PeerMe	?	Proprietario freeware
PGPfone (superseded by Zfone)	?	Codice visibile + licenza proprietaria
PJSIP SIP Stack	Linux, Win32 (NT/200/XP), WinCE/Windows Mobile, MacOS X, FreeBSD, RTEMS, Symbian, etc	GPL Free software
PhoneGaim	Linux (Linspire), Windows XP/2000	GPL Free software
Phoner	Windows	Proprietario freeware
PhonerLite	Windows	Proprietario freeware
ReSIProcate	?	Vovida Software License (open source)
SightSpeed	Mac / Windows	Proprietario freeware
SIMPLE	?	
SIP Communicator	Linux, Mac, Windows XP/2000 (all java supported)	GNU Lesser General Public License
SIP Express Router (SER)	?	
sipX	?	Open source
SJphone	MS Windows, Mac, Linux, Win Mobile 5.0	Proprietario
Skype	Windows XP/2000, Mac, Linux, Pocket PC	Proprietario/chiuso
***starShop**	?	Open source
Tapioca	Linux	GNU Lesser General Public License
TelTel	Windows	Proprietario

TERAVoice Server	?	
Tivi	?	
Twinkle	Linux (especially KDE)	GPL Free software
Vbuzzer	?	Proprietario freeware
VoipDiscount	?	Proprietario freeware
WengoPhone	Linux, Mac, Windows XP/2000	GPL Free software
Windows Live Messenger	Windows XP/2000	Proprietario freeware
Yahoo! Messenger	Windows, Mac	Proprietario
X-Lite	Windows, Mac, Linux	Proprietario
YATE	Windows, Linux	GPL/MPL Free software
Zfone	Windows, Linux, Mac	Codice visibile + licenza proprietaria

Nella tesi si utilizzerà il softphone SJPhone, per la comunicazione via SIP e H.323, e per la videochiamata il softphone Kapanga e Xlite.

» 2.12 Asterisk vs Skype

Può essere utile confrontare, seppure forse in maniera non approfondita, gli aspetti salienti di Asterisk con il più famoso e diffuso sistema VoIP Skype.

Skype ha dalla sua parte il pregio di essere stato il primo sistema per il VoIP a diffondersi a larga scala, non ha bisogno di configurazioni (nemmeno per superare NAT o firewall), è immediato da usare, ha già versioni del client per tutte le piattaforme più diffuse, dai PocketPC agli Smartphone, ha funzionalità come l'invio di sms ai cellulari.

Lo svantaggio di Skype è la scarsa flessibilità (un conto è configurare router e firewall una volta per tutti i dipendenti di un'azienda, un'altra cosa è gestire 100 o più account separati, ognuno col proprio credito, ecc.), ma soprattutto il fatto che, a differenza di Asterisk, non è open source, ma offre un servizio chiuso e proprietario e ciò significa che

solo gli apparecchi direttamente supportati da Skype possono funzionare con esso.

Dunque Skype non potrà mai aspirare a diventare uno standard, se non rivelerà il suo funzionamento, a tutto vantaggio di sistemi gratuiti e aperti a tutti come appunto Asterisk e i protocolli SIP e IAX.

» 2.13 Importanza di Asterisk

Asterisk è importante per molti motivi:

- **Estrema riduzione dei costi** – combinato con un hardware telefonico Linux a basso costo, Asterisk può essere utilizzato per creare PBX ad una frazione del prezzo dei tradizionali PBX, mentre è provvisto di funzionalità maggiori di gran parte dei sistemi disponibili;
- **Controllo** – Asterisk permette all'utente di avere il controllo del proprio sistema telefonico. Quando una chiamata è nel Linux box, con Asterisk è possibile fare qualsiasi cosa. Infatti la sua natura Open Source gli conferisce grande scalabilità e controllo nelle operazioni, candidandolo a diventare un prodotto che sorpasserà la fetta di mercato dei concorrenti proprietari.
- **Rapida creazione e sviluppo** – Asterisk permette alle applicazioni PBX e IVR di essere rapidamente create e sviluppate. La sua potente Command Line Interface (CLI) e la scrittura nei file di testo permettono una rapida configurazione e diagnostiche in tempo reale.
- **Ricche e ampie features di base** – Non solo Asterisk è Open Source e sviluppa via software servizi molto costosi con sistemi proprietari, come voicemail, menù voce, IVR e servizi di conferenza, ma permette anche l'aggiunta di nuove features con rapidità e minimo sforzo;

- **Personalizzazione** – Attraverso un supporto internazionale, la semplice configurazione dei file e la modifica del codice sorgente, ogni aspetto di Asterisk può essere personalizzato. Per esempio, i codici per le features delle chiamate possono essere modificati per confrontarli con i sistemi esistenti.
- **Creazione dinamica del contenuto** – Allo stesso modo in cui un server web permette all'utente una creazione dinamica del contenuto (es. consultazione degli orari dei treni, aggiunta di un post in un forum, modifica di password personale, ecc.), Asterisk permette di creare contenuto dinamico attraverso il telefono;
- **Dialplan estremamente flessibile** – Il dialplan di Asterisk permette l'integrazione delle funzionalità di un PBX e un IVR. Molte delle features esistenti di Asterisk (e quelle desiderate per il futuro) possono essere implementate utilizzando niente più che estensioni logiche, che non hanno un vincolo sulla lunghezza.