5.1 HARD

5.1.1 Diagrama en bloques

Las PBX virtuales no emplean conmutación de circuitos, sino conmutación de paquetes. Con lo cual el diagrama en bloques del presente proyecto es el siguiente:



5.1.2 Descripción detallada de cada bloque

Placa Principal:

Este bloque es el encargado de interconectar todos los demás componentes de la PBX. En él están presentes la memoria, los slots de expansión donde se conectan las tarjetas, diversos chips de control como el del BIOS y los slots donde se conectan los microprocesadores. Los paquetes de datos pasan siempre por este bloque pero no es el encargado de realizar la conmutación de los mismos.

Unidad de almacenamiento:

El código fuente del software y todo el ambiente de configuración de la PBX junto con otras configuraciones van incluidos en esta unidad. No se requiere que sea muy voluminosa pero si debe tener una interface rápida (por ejemplo la SCSI) para que el microprocesador pueda realizar la conmutación de paquetes de manera eficaz.

Microprocesador:

Este es el bloque más importante debido a que es el encargado de realizar todas las tareas de procesamientos de la PBX además de la conmutación de los paquetes como se menciono antes. Considerando que en la facultad habrá aproximadamente cuatrocientos cincuenta internos el microprocesador debe ser el más potente que se pueda conseguir en el mercado.

Fuente de alimentación:

Se contara con al menos dos fuentes de alimentación para la PBX, una que se encargue de alimentar a todos y cada uno de los módulos del equipo, y otra de resguardo. La potencia de las fuentes debe ser de 65WATTS.

Placa TDM:

Este modulo es la interfaz entre la PBX virtual y las líneas analógicas que se deseen conservar en la facultad. Se asume que este número de líneas va a ser inferior a ocho debido a que no es recomendable usar más de dos tarjetas de comunicaciones en este sistema y la máxima cantidad de puertos FXO disponibles para esta tarjeta es de ocho. También disponen de puertos FXS que corresponden a las extensiones analógicas.

Placa E1:

Este bloque es la interface digital entre la PBX virtual y el “router” de la empresa proveedora de telefonía digital para la facultad. Evidentemente el enlace contratado debe corresponderse con la norma E1 de treinta canales de comunicación.

Placa de red:

Esta interfaz es el nexo que conecta la PBX virtual a la red de la facultad por la cual trasmitirá todos los paquetes de voz.

De todo lo anterior se concluye que la solución perfecta que encaja con el diagrama anterior es un servidor de alto porte con las tres placas descriptas anteriormente.

5.1.3 Detalles de selección y calculo de los elementos circuitales de cada bloque

Como se menciono en el párrafo anterior, se debe partir de una base de un servidor robusto que pueda realizar el procesamiento de la conmutación de paquetes para aproximadamente cuatro ciento cincuenta internos. Obviamente no se van a conmutar los cuatro ciento cincuenta internos al mismo tiempo, sin embargo se debe contar con el equipo mas potente que se pueda conseguir. Puntualmente debe tener al menos dos Procesadores Quad Core y una memoria RAM de al menos 4GB. La placa de red debe ser de GigaBit y, en lo concerniente a las placas TDM y E1, estas deben elegirse de acuerdo a la cantidad de líneas analógicas y digitales de las que dispone la Facultad. Se recomiendan “arreglos” de discos para disponer de redundancias ante la falla de alguno y de capacidad de 80GB para alojar tanto el software de la PBX como la base de datos que se encargara de registrar todas las llamadas (esto ultimo se recomienda que en el mediano plazo se migre a otro equipo para que no le reste rendimiento a la PBX). Todo esto a confeccionarse de manera ordenada en un pliego.

5.1.4 Plan de pruebas de cada modulo

1. Actuación de cualquiera de las dos fuentes ante la falla de alguna, o el corte de suministros de la fase donde se encuentra conectada.
2. Simulación de falla de los discos para probar el “ARRAY”.
3. Testeos de hardware (placa mother, microprocesador, discos rígidos, fuentes de alimentación y Placas de red y de telefonía) mediante la ejecución de comandos de la plataforma y/o ejecución de scripts propios o del fabricante.
4. Rápida conmutación del servidor de contingencia ante una falla catastrófica del servidor de producción.
5. Control de baja latencia y buena calidad de sonido mediante una llamada entre internos de la facultad.
6. Garantía del ancho de banda adecuado para asegurar el punto anterior mediante la prueba de diferentes “CODECS”.
7. Prueba en “masa” de varias llamadas al mismo tiempo para corroborar los dos puntos anteriores, en horarios “picos” de máximo trafico de la red.

5.2 SOFT

5.2.1 Diagrama de Estados, procesos y flujogramas

5.2.1.1 Encapsulamiento de una trama de VoIP



Por ejemplo, si el CODEC usado es G.711 y el periodo de paquetización es 20 ms, la carga útil será de 160 bytes. Esto resultara en una trama total de 206 bytes en una red WAN y en 218 bytes en una red LAN.

5.2.1.2 Ciclo de una llamada



5.2.1.3 Recorrido de la llamada



5.2.1.4 IVR (interactive Voice Responce)



5.2.2. Descripción de subrutinas

Como se ha mencionado en reiteradas oportunidades, el lenguaje utilizado para la aplicación de este proyecto es “libre”. El código fuente esta basado en un lenguaje de scripts de asterisk que es de fácil entendimiento para cualquier administrador de redes.

Nota: A modo de sugerencia se ejemplifica a continuación los archivos en donde deben ir configurados los scripts del lenguaje del PBX de los gráficos anteriores pero de ninguna manera esto es concluyente.

De acuerdo a los diagramas vistos anteriormente se debe partir del “dialplan” o plan de discado, el cual debe ser definido en pos del mejor entendimiento por parte de todos los usuarios de telefonía. Una vez definido el plan de discado, se deben volcar todos los internos a un archivo de configuración denominado “extensions.conf” mediante el respectivo lenguaje de asterisk. En este mismo archivo se puede configurar el ivr (o respuesta interactiva de voz). El scripts básico para la configuración de cada uno de los internos es el siguiente:

exten=***nro\_int***,***nro\_sec***,Dial(**prot**/***username***,***timeout***,tr)

Por ejemplo, para un tiempo de espera de 20 segundos, luego se active el mensaje de voz y por ultimo el tono de desconectado, la configuración sería la siguiente:

[internos]

exten=1000,1,Dial(SIP/Pablo,20,tr)

exten=1000,2,Voicemail(1000)

exten=1000,3,Hangup

La primer línea se observan los “[]” que significan el grupo de llamadas que se pueden realizar. De esta forma se puede clasificar en llamadas internas, externas, de larga distancia, nacionales, internacionales, etc.

Los mensajes de voz o “voicemail” para los usuarios se deben configurar en un archivo denominado “voicemail.conf”, cuya sentencia básica es la siguiente:

***Nro\_int*** => ***nro\_int***,***username***,***direccion de mail***

Continuando con la línea del ejemplo anterior, la configuración del voicemail para el usuario pablo sería la siguiente:

[internos]

1000 => 1000,pablo,pablo@fi.uba.ar

Se concluye, como se puede apreciar de los ejemplos anteriores, que los comandos para la administración y configuración del sistema son fáciles e intiutivos. Es decir, con un comando script se pueden configurar miles de internos, música en espera, conferencias, grabación de llamadas, tarifación, etc. Aquí esta la justificación de una consola grafica para la administración del sistema.

5.2.3. Plan de prueba de módulos y de depuración de soft

La ayuda para depurar más sencilla es por supuesto la interfaz de comandos de asterisk. Existen otros mecanismos para depurar código como por ejemplo el seguimiento de LOGS pero la interfaz que se describirá a continuación es el mecanismo por excelencia para el seguimiento del código del soft.

Asterisk soporta un intérprete de comandos (CLI: Command Line Interface), del estilo de muchos routers. Para ingresar a esta consola se debe ejecutar el comando asterisk –r. Este intérprete es bastante potente y permite controlar y monitorear la PBX. Soporta el empleo de la tecla <tabulador> para ver un listado de todos los comandos disponibles, completar un comando o argumento, o ver posibles argumentos. También se pueden ejecutar comandos sin estar dentro del intérprete pero a los efectos de debugguear código se debe estar conectado.

Conexión a la interfaz de comandos:

asterisk -r

Verificar versión actual de Asterisk:

CLI> core show version

Verificar el tiempo que lleva de ejecución el Asterisk:

CLI> core show uptime

Recargar la configuración:

CLI> reload [modulo]

Detener el servicio:

CLI> stop now | gracefully | when convenient

Salir de la interfaz de comandos:

CLI> quit

Verificar usuarios SIP registrados:

CLI> sip show peers

CLI> sip show users

Verificar canales activos

CLI> core show channels

Verificar usuarios IAX registrados:

CLI> iax2 show peers

CLI> iax2 show users

Verificar dialplan

CLI> dialplan show

5.2.3.1 Verbose

Nivel de “Verbose”: Este valor indica la cantidad de mensajes que se recibirán sobre los eventos generales de la secuencia de una llamada. Cuanto más alto se recibirá más información sobre lo que sucede en la PBX.

Este nivel, se puede establecer de varias formas:

Al arrancar el demonio:  
asterisk -vvvvvv

Al conectarse al demonio:  
asterisk -rvvvvvvvv

Desde la interfaz de comandos:  
CLI> core set verbose 5

5.2.3.2 Debug

Nivel de “Debug”: Este valor indica la cantidad de mensajes que se recibirán sobre los eventos generales del sistema, pero se utiliza en general para depurar problemas de drivers o de aplicaciones.

Este nivel, se puede establecer de varias formas:

Al arrancar el demonio:  
asterisk -dddd

Al conectarse al demonio:  
asterisk -rdddd

Desde la interfaz de comandos:   
CLI> core set debug 5

Para debugear podemos ajustar el nivel de verbose y debug de la consola con los siguientes comandos

extranet\*CLI> core set debug 3

Core debug was 0 and is now 3

extranet\*CLI> core set verbose 3

Verbosity is at least 3

extranet\*CLI> core set debug 0

Core debug is now OFF

Para un debug mas preciso podemos debugear directamente en el canal, por ejemplo con SIP podemos hacer lo siguiente

extranet\*CLI> sip set debug ip 80.58.3.33

SIP Debugging Enabled for IP: 80.58.3.33

extranet\*CLI> sip set debug off

SIP Debugging Disabled

Podemos ver los usuarios registrados en el momento con sip show peers, los usuarios que aparecen con Host (Unspecified) están cacheados pero ya no están registrados.

extranet\*CLI> sip show peers

Name/username Host Dyn Nat ACL Port Status Realtime

xxxxxxxxx/xxxxxxxxx (Unspecified) D N 0 Unmonitored

xxxxxxx/xxxxxxx (Unspecified) D N 0 Unmonitored

xxxxxxx/xxxxxxxx xx.xxx.xx.118 D N 62331 Unmonitored

xxxxxxx/xxxxxxx xx.xx.x.82 D N 52106 Unmonitored

xxxxxxxxxxxxxxx (Unspecified) D N 0 Unmonitored

xxxxxxxx/xxxxxxxx xxx.xx.xxx.232 D N 5060 Unmonitored

xxxxxxxxxxx/xxxxxxxxxxxxx (Unspecified) D N 0 Unmonitored

7 sip peers [Monitored: 0 online, 0 offline Unmonitored: 3 online, 4 offline]

También podemos administrar las salas de conferencia con el comando meetme, ver quienes están conectados a una sala o expulsar a alguien

extranet\*CLI> meetme list 52

User #: 01 1188 xxxxxx Channel: SIP/xxxxxx-083a7ef0 (unmonitored) 00:00:04

1 users in that conference.