UNIVERSIDAD CATÓLICA ANDRÉS BELLO
Urb. Montalbán - La Vega - Apartado 29068
Teléfono: 407-4493 Fax: 407-4590
Caracas, 1021 - Venezuela
Facultad de Ingeniería
Escuela de Ingeniería de Telecomunicaciones

ANALISIS TECNICO DE LA RED DE DATOS Y TELEFONICA DE LA UCAB PARA LA IMPLEMENTACION DE LA TECNOLOGIA DE VOZ SOBRE IP

APENDICES

REALIZADO POR: Luis Fernando Ascanio

TUTOR: Prof. Victor Nesterovsky

Caracas, 28 de febrero de 2008

Apéndice A Manual de la herramienta Chariot

Qué es Chariot?

Chariot es una herramienta que permite generar tráfico de forma controlada y planificada, con la finalidad de emular aplicaciones sobre redes de datos en tiempo real, sin la necesidad de instalar tales aplicaciones en la plataforma de red. Permíte estudiar, analizar y predecir de manera acertada el comportamiento de ciertas aplicaciones en una infraestructura de red determinada.

Utilizando este *software*, es posible:

1. Probar el rendimiento y la capacidad de una infraestructura de red, a nivel de hardware y *software*.
2. Realizar análisis comparativos de productos antes de adquirirlos.
3. Identificar la causa de posibles problemas en la implementación de nuevas tecnologías sobre la plataforma.
4. Predecir los efectos sobre la infraestructura de red, que tendría implementar una u otra tecnología.
5. Verificar el rendimiento esperado para ciertas aplicaciones críticas, que poseen requerimientos especiales para su correcto funcionamiento como en el caso de voz y video.

Funcionamiento

Para poder realizar pruebas con el *software* Chariot, es necesario establecer una topología determinada. Deben existir los siguientes equipos:

- Una Consola: Es el equipo donde se encuentra el programa principal, donde se programan y establecen todos los parámetros de la prueba, y donde se reciben los resultados de las mismas.
- Dos, o varios Performance *Endpoints*: Es un servicio, que se encuentra activo en un computador, bien sea un servidor, *laptop* o PC. El equipo sobre el cual se instala y funciona el Performance *Endpoint*, debe ser un equipo

suficientemente robusto, donde preferiblemente no se estén llevando a cabo procesos adicionales que dediquen el procesamiento del equipo a otras tareas.

La persona encargada de programar las pruebas, realiza la configuración de las mismas desde la consola, estableciendo todas las características de la misma: cuáles serán los *endpoints* involucrados, el tipo de tráfico que se generará, etc. Los equipos involucrados en la prueba deberán tener instalados el servicio de *endpoints*, lo que permitirá la correcta operación de la aplicación.

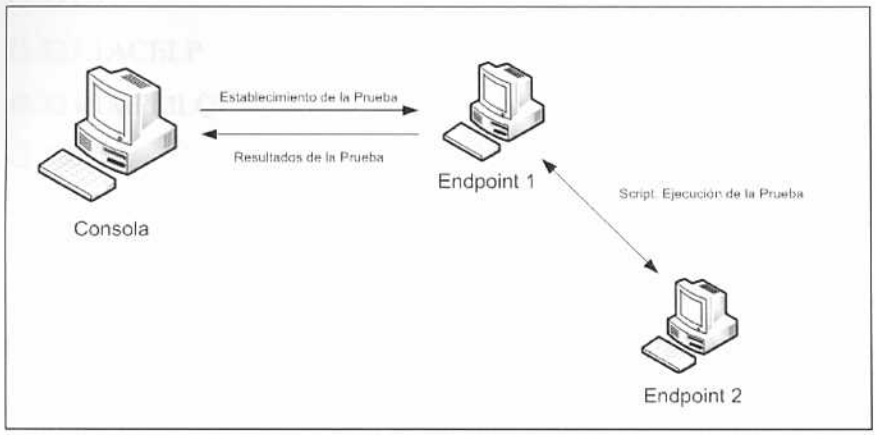


Figura 1: Representación de los componentes que conforman la herramienta Chariot de NetIQ Chariot. (Autor)

La función principal de los *endpoints* es llevar a cabo la prueba, y el *endpoint* 1 es el encargado de recolectar y reportar los resultados de la misma a la consola. Un "par"de *endpoints*, concepto bajo el cual la aplicación describe y configura las pruebas, constituye básicamente de dos direcciones IP que corresponden a los *endpoints*, el protocolo que se utilizará en la prueba, y el tipo de aplicación que se pretende emular. Una prueba puede incluir un único par, o incluso cientos o miles de ellos, todo depende de la complejidad y los alcances de la misma; así como de las características de la licencia obtenida por el fabricante.

Para cada prueba se selecciona un *script*, que es el que contiene todos los parámetros que le indicaran a cada uno de los *endpoints* las acciones que deben llevar a cabo en cada prueba, los cuales se llamarán de ahora en adelante E1 y E2 respectivamente. El *software* permite igualmente la modificación de dichos *scripts* para fines específicos aunque se recomienda solo para usuarios con cierto nivel de experiencia y

conocimientos. La consola de Chariot, viene equipada con un conjunto preestablecido de *scripts* que permiten emular las aplicaciones más comunes existentes en una plataforma de red, entre los más relevantes para la realización de la investigación se encuentran:

- Troguhput
- Response Time
- G.711A
- G.711U
- G.723.1ACELP
- G.723.1MPMLQ
- G.726
- G.729

Particularmente los *scripts* que permiten emular aplicaciones de VoIP, se agrupan en un modulo definido por la aplicación como VoIP Test Module. Para establecer una prueba de VoIP, deben tomarse las siguientes acciones:

1. Seleccionar en la consola Chariot, la opción New…
2. Se abre una nueva ventana, donde deberán incluirse los pares que se deseen incluir en la prueba.

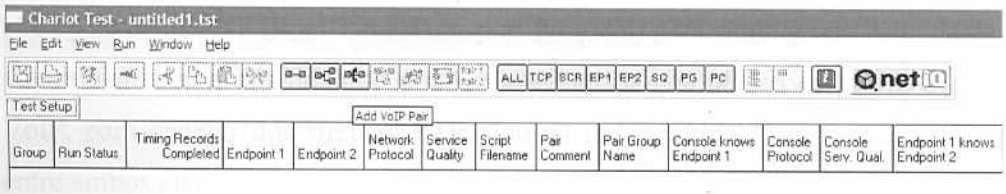


Figura 2: Agregando un par de VoIP desde la consola de Chariot. (Autor)

3. Al agregar un par, aparecerá la siguiente ventana, donde deben colocarse todos los parámetros de la prueba: Dirección IP del E1, Dirección del E2, así como el *script* que se pretende utilizar en la comunicación. En este *script* se selecciona el tipo de codec, si se utilizará o no la técnica de supresión de silencios, así como el porcentaje de actividad, establecer políticas de calidad de servicio, entre otros.

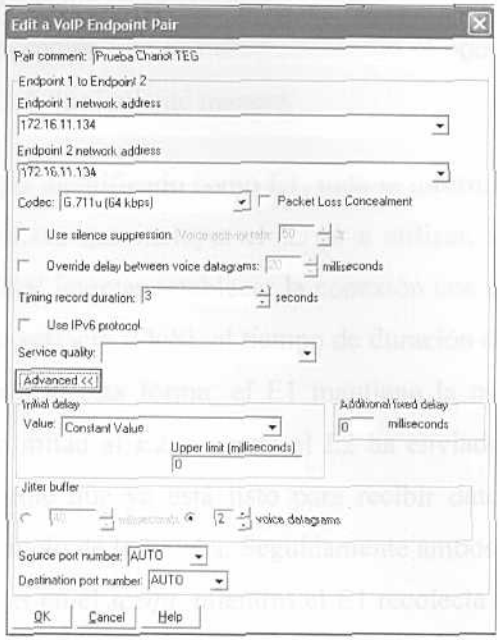


Figura 3: Configurando las propiedades de un par de VoIP desde la consola de Chariot. (Autor)

Ahora bien, si se seleccionan opciones avanzadas, se pueden modificar ciertos parámetros importantes en la configuración de las pruebas, como lo es el delay inicial o Inicial Delay. Este valor es muy importante, ya que permite que en una misma prueba, se incluyan flujos adicionales, para observar así el rendimiento de la plataforma al incrementar la cantidad de tráfico.

Es importante mencionar, que las pruebas de voz realizadas con Chariot, generan un stream de datos RTP (Real Time Transport Protocol) para simular una llamada de VoIP. Ahora bien, con la finalidad de simular una llamada real, deben generarse dos pares de VoIP, para generar dos streams RTP, uno en cada sentido. Estos steams RTP fluyen entre ambos *endpoints*, y son los que permiten calcular el rendimiento de la plataforma de red para servicios de VoIP, logrando hasta cierto punto predecir la percepción que tendrá un usuario promedio de dicho servicio. En este documento, y con fines ilustrativos se colocó un único par.

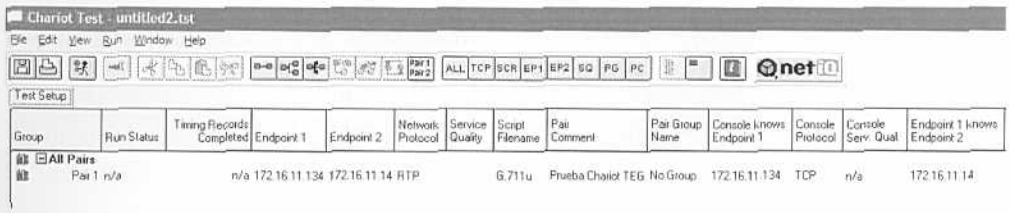


Figura 4: Iniciando una prueba de VoIP desde la consola de Chariot. (Autor)

4. Al hacer clic en "Run..." se inicia la ejecución de la prueba, mediante un proceso de comunicación que es transparente para el operador de la prueba, que puede ser descrito de la siguiente manera:

La consola envía al equipo identificado como E1, toda la información requerida para el establecimiento de la prueba, que incluye: el *script* a utilizar, la dirección del E2, el protocolo que se utilizará al intentar establecer la conexión con el E2, si se utilizarán o no políticas de calidad de servicio (QoS), el tiempo de duración de la prueba, y cómo se reportarán los resultados. De esta forma, el E1 mantiene la mitad del *script* para sí mismo, y reenvía la otra mitad al E2. Cuando el E2 ha enviado una confirmación de recepción al E1 indicándole que ya está listo para recibir datos, el E1 notifica a la consola y ésta ordena el inicio de la prueba. Seguidamente ambos *endpoint*s ejecutan las instrucciones especificadas en el *script*, mientras el E1 recolecta todos los registros que contienen los resultados de la prueba. Al finalizar el proceso, el E1 envía todos los registros a la consola, la cual se encargará de interpretar, analizar, y mostrar los resultados al operador de una manera coherente.

5. Luego de culminada la prueba, se obtiene una cantidad de información importante desplegada en una serie de pestañas, que permitirá realizar el estudio del comportamiento de la aplicación seleccionada sobre la plataforma de red sobre la cual se está realizando la prueba.

Con el módulo de VoIP particularmente, se obtiene una cantidad de información importante, la cual puede visualizarse navegando a través de estas pestañas. A continuación se describen cada una de esas pestañas, la información que contienen, así como el procedimiento de cálculo que realiza la aplicación para lograr la graficación de esta información; y de esta forma ilustrar al lector en detalle sobre el funcionamiento de la aplicación.

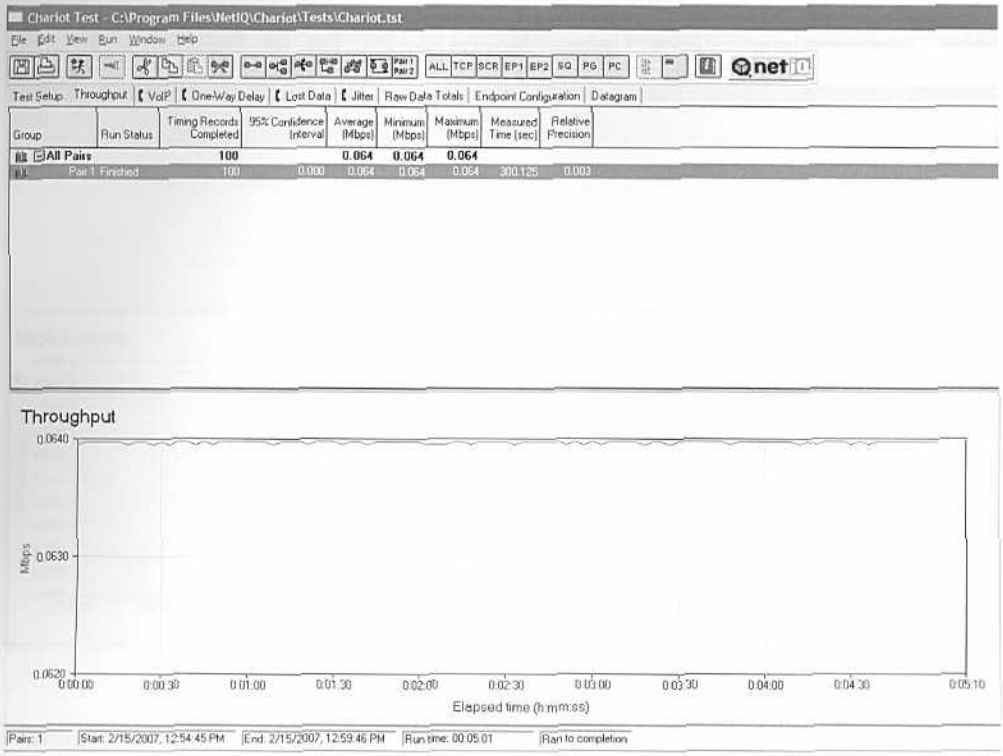


Figura 5: Resultados obtenidos al finalizar una prueba de VoIP. (Autor)

La primera pestaña que se presenta, muestra resultados acerca del *throughput* obtenido a lo largo de la ejecución del *script*.

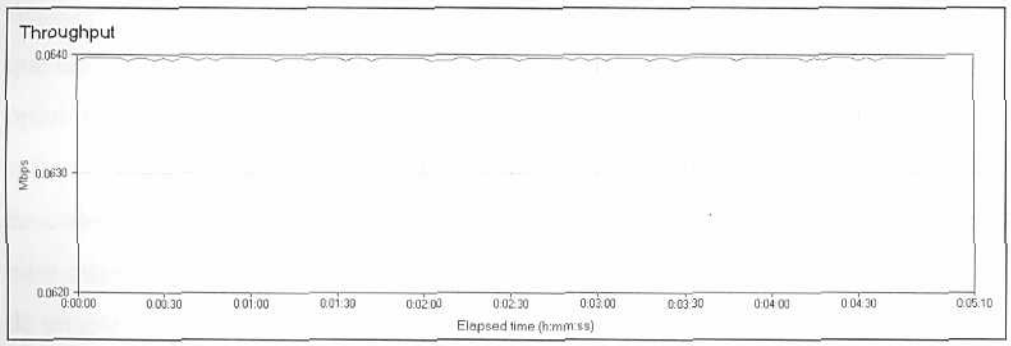


Figura 6: Gráfica de *Throughput* obtenida con Chariot. (Autor)

La formula utilizada para obtener el valor de *throughput* es la siguiente:

T = (Bytes Recibidos por E2) / (Tiempo de Prueba)

Seguidamente se observa la pestaña llamada VoIP, como se muestra a continuación.

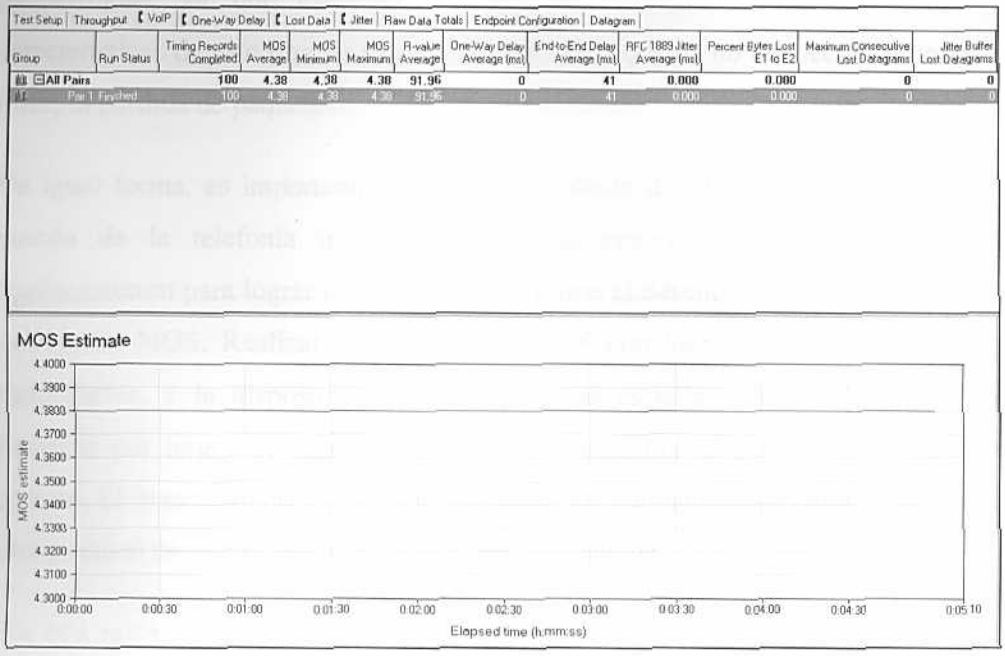


Figura 7: Valor estimado de MOS obtenido con Chariot. (Autor)

La información mostrada es un estimado del valor de MOS (Mean Opinión Store) objetivo, y representa un indicador de la calidad relativa que tendrá una llamada. Chariot utiliza una versión modificada de las ecuaciones propuestas por el estándar de la ITU G.107 para realizar este cálculo de MOS.

Anteriormente, para conocer el nivel de calidad para una llamada se realizaban pruebas que eran siempre de tipo subjetivas, es decir, se tomaba el teléfono y un usuario daba su opinión en cuanto a la calidad de la voz. La medida más utilizada para establecer valores definidos en cuanto a la calidad de la voz es el valor de MOS, el cual se describe en el Standard ITU P.800. En los últimos tiempos, se han realizado esfuerzos para establecer técnicas objetivas para medir la calidad de una llamada, y el desarrollo de un grupo de estándares, demuestra estos avances:

- PSQM (ITU P.861) / PSQM+ - Perceptual Speech Quality Measure
- MNB (ITU P.861) - Measuring Normalized Blocks
- PESQ (ITU P.862) - Perceptual Evaluation of Speech Quality
- PAMS (British Telecom) - Perceptual Analysis Measurement System

Sin embargo este grupo de avances, no se adaptaron (ya que no fueron diseñados con ese fin) para asegurar o establecer una medida de la calidad de la voz sobre redes

paquetizadas, ya que estos no conocen ni manejan el funcionamiento y las características de este tipo de redes. Este tipo de modelos no conocen acerca de delay, *jitter*, ni pérdida de paquetes.

De igual forma, es importante saber que el cálculo de MOS como tal, proviene del mundo de la telefonía tradicional y es un criterio que todos los vendedores implementaron para lograr mapear de cierta forma el desempeño de sus equipos con los valores de MOS. Realizar mediciones de MOS con humanos, requiere una muestra significativa, y la disposición de los mismos a escuchar continua y repetidamente llamadas por largos periodos de tiempo, lo que es por lo general significativamente costoso. El Standard ITU P.800 describe cómo los humanos reaccionan y que puntaje le otorgarían al pretender medir la calidad de una llamada.

Por esta razón se introdujo el modelo-E o E-model, el cual es un modelo desarrollado por la ETSI (European Standards Institute), que posteriormente se convirtió en la recomendación ITU G.107. Este modelo permite evaluar la calidad de una transmisión, tomando en cuenta todos aquellos factores intrínsecos de una red paquetizada que incluyen directamente en la degradación de la información de voz, sobre una plataforma de red. Este modelo logra establecer una correspondencia entre las características de la red, y los puntajes de acuerdo a la calidad percibida, hacen que estas estimaciones de MOS obtenidos objetivamente sean realmente significativos, cuando se busca determinar el rendimiento de la voz sobre redes paquetizadas. El modelo calcula un valor conocido como valor-R o R-value, el cual permite hacer una correlación del cálculo objetivo, obtenido por una serie de parámetros calculados, directamente con un valor en la escala MOS.

Chariot realiza modificaciones leves al modelo-E, al utilizar los siguientes factores para calcular el valor-R, y de allí obtener un estimado objetivo de MOS. Cada una de estas pestañas se observan de forma separada en los resultados proporcionados por la consola, de manera tal que el operador pueda saber cual de las características de la red pueda estar causando la disminución de la calidad de las llamadas en la plataforma. A continuación se describe brevemente de que manera se realiza cada una de esas mediciones:

- Retardo o Delay: Este retardo representa el tiempo que le toma a un datagrama RTP viajar desde el E1 a E2. Incluye únicamente el tiempo de propagación. Cada vez que E1 envía un datagrama le adhiere una marca de tiempo o time stamp que indica el tiempo exacto en el que dicho datagrama fue enviado. De esta forma cuando E2 recibe el datagrama, y conocido el tiempo en el que lo recibió y el tiempo en el que fue enviado, puede saber el tiempo que le tomo al datagrama viajar de un *endpoint* a otro.

$$\text{Delay} = T2 - T1.$$

Donde: T2 = Tiempo de recepción T1= Tiempo de envío.

Con la finalidad de realizar mediciones acertadas, antes de iniciar la transmisión de los streams RTP, los *endpoints* sincronizan sus relojes para asegurar un cierto nivel de confiabilidad en las mediciones, por lo general utilizando el reloj interno de la maquina donde se instaló el *endpoint*.

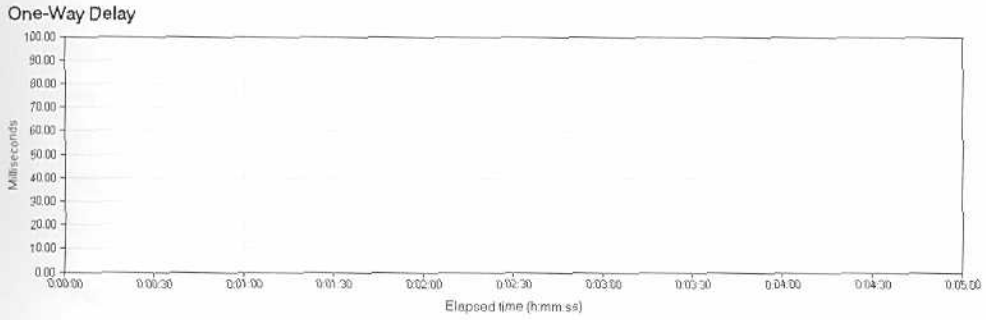


Figura 8: Gráfica de retardo en un sentido, obtenida con Chariot. (Autor)

- Retardo total o End-to-End Delay. Incluye los retardos asociados a otros factores importantes como el esquema de paquetización, el tamaño de los buffers, etc.
- Packetization Delay. Asociado a la conversión de la señal de voz de analógica a digital. Varía según el tipo de codec utilizado.

Codec	Bit rate (kbps)	Frame time (ms)	Look ahead (ms)	Codec impairment
G.711	64.0	10	0	0
G.729	8.0	10	5	11
G.723.1-MPMLQ	6.3	30	7.5	15
G.723.1-ACELP	5.3	30	7.5	19
G.726	32.0	10	0.125	7

Figura 9: Características conocidas de los codec utiliados por Chariot para la realización de las pruebas. (Fuente: Ixia Company. http://www.ixiacom.com/library/white_papers/display?skey=voip_quality)

- Porcentaje de Paquetes Perdidos o Data Loss Percentage. Número total de datagramas perdidos.

$$L = Br - Be$$

Donde: Br = Bytes_Recibidos Be = Bytes_Enviados

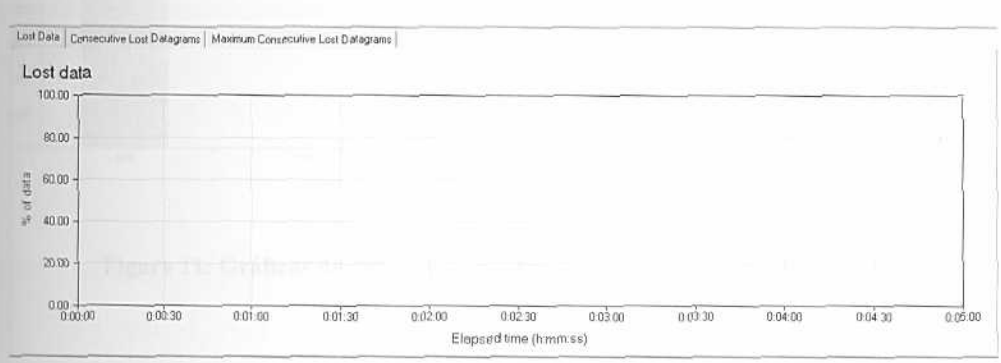


Figura 10: Gráfica de Porcentaje de paquetes perdidos obtenidos con Chariot. (Autor)

De igual forma, la herramienta permite conocer qué cantidad de datagramas perdidos, se perdieron de forma consecutiva, así como la cantidad máxima de datagramas perdidos.

- Valor de *Jitter* Promedio: La aplicación es capaz de calcularlo de dos formas: mediante las especificaciones del RFC1889, o calculando directamente la variabilidad del retardo. Cuando se calcula el *jitter* utilizando el RFC1889, lo que se busca es obtener la media de las diferencias, que llamaremos D, entre los intervalos de tiempo en que E1 envía un par de datagramas, y E2 los recibe. Esto equivale exactamente a la diferencia entre la marca de tiempo de un datagrama RTP y el momento de su recepción por el E2. Si "Ti" representa la marca de tiempo de un datagrama llamado I, y a su vez "Ri" es el tiempo en el

que llego dicho datagrama, entonces para dos datagramas J e I, se expresa esta diferencia como

$$D(i,j)=(Rj-Ri)-(Tj-Ti)=(Rj-Tj)-(Ri-Ti)$$

El *jitter* es calculado entonces para cada nuevo datagrama recibido, tomando en cuenta el último datagrama recibido y así consecutivamente.

$$J=J+(|D(i-1,i)|-J)/16$$

Por otra parte, el *jitter* expresado como variación de retardo, tal y como lo plantea la aplicación, se refiere únicamente al porcentaje de datagramas de la comunicación que experimentaron variaciones en el retardo.

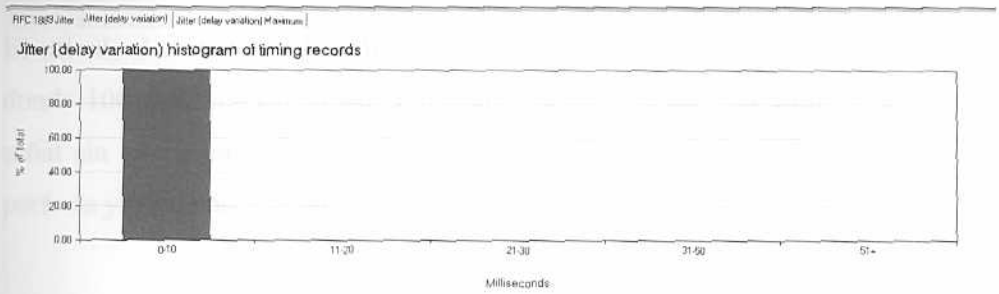


Figura 11: Gráficas de *Jitter* (histograma) obtenidas con Chariot (Autor)

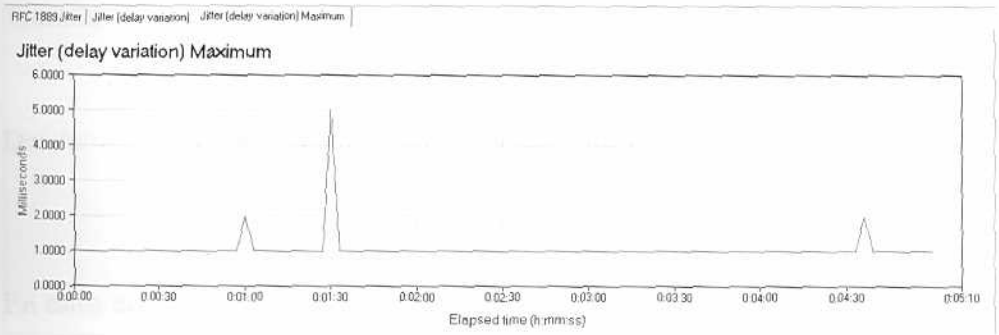


Figura 12: Gráficas de *Jitter* (variación de delay) obtenidas con Chariot (Autor)

Finalmente esta pestaña concluye especificando cuál fue la máxima variación de retardo en milisegundos, experimentada por esos datagramas mencionados anteriormente.

La tabla que se presenta a continuación, resume la relación entre los valores mínimos de MOS objetivo que se obtienen al realizar pruebas con el *software* Chariot, necesarios para alcanzar un cierto grado de satisfacción entre un grupo de usuarios.

Mean Opinion Score MOS (límite inferior)	Satisfacción del usuario
4.34	Muy satisfecho.
4.03	Satisfecho.
3.60	Algunos usuarios no satisfechos
3.10	Muchos usuarios no satisfechos
2.58	Casi todos los usuarios no satisfechos

Figura 13: Correspondencia de los valores de MOS obtenidos con la satisfacción de los usuarios. (NetIQ Chariot. Help Library)

El cálculo de este valor-R antes mencionado, puede tomar valores desde o hasta 100, donde 100 significa excelente, y 0 deficiente. El cálculo del valor-R se inicia con la señal sin alteraciones. Si no existen equipos de red, puede decirse que la calidad es perfecta y puede decirse que:

$$R=Ro$$

Pero este escenario representa condiciones ideales que no existen en ninguna plataforma de red. Ya que sí existen equipos de red que modifican la calidad de la senal, degrandándola mientras ésta viaja a través de la red, entonces podemos decir que:

$$R=Ro - Id - Ie$$

Donde: Id = Retardo total o End-to-end Delay

Ie = Retardo introducido por los equipos de red.

En estos dos grupos de retardo, es donde se encuentran los factores de Retardo, *jitter*, y paquetes perdidos, además del esquema de paquetización del codec seleccionado ya que cada uno distorsiona de una manera particular la señal original de voz. A continuación se muestra la correspondencia entre diferentes valores-R obtenidos, y su correspondencia con un nivel de MOS.

Figura 14: Correspondencia entre valor-R - MOS. (Fuente: Ixia Company. http://www.ixiacom.com/library/white_papers/display?skey=voip_quality)

De por si el mayor valor de R que puede obtenerse al utilizar Chariot es de 93.2, lo que corresponde a un valor de MOS 4.41.

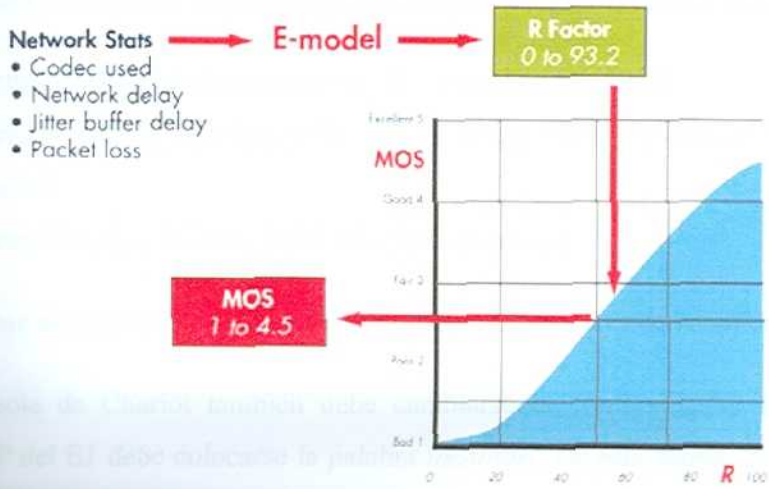


Figura 15: Funcionamiento del E-model para el cálculo del valor estimado de MOS.

http://www.ixiacom.com/library/white_papers/display?skey=voip_quality)

En definitiva, lo que permite el modelo-E, es tomar todos esos parámetros que pueden calcularse fácilmente, provenientes del desempeño propio de la red, traducir estos valores a un determinado valor-R y de allí obtener un estimado objetivo del MOS.

Configuración para entornos con NAT (Network Address Translation)

Cuando existe en la comunicación algún equipo encargado de realizar funciones de NAT, la herramienta debe ser configurada para estos casos. De igual forma, debe

configurarse debidamente el equipo que está realizando NAT, de forma tal que permita la comunicación sin problemas entre los *endpoints*.

La configuración necesaria en el equipo encargado de hacer el NAT bien sea un *firewall* o un *router*, debe incluir, como se ilustra en la siguiente figura:

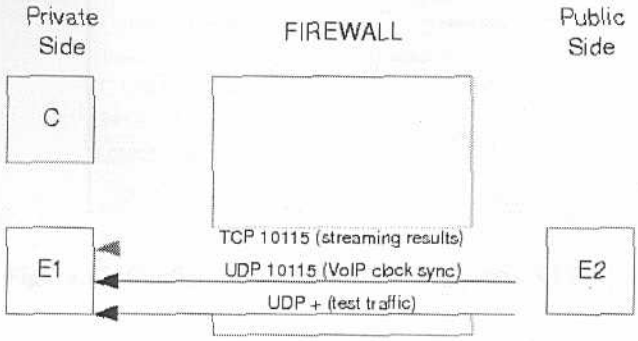


Figura 16: Configuración necesaria bajo entorno con NAT. (NetIQ Chariot. Help Library)

- Permitir la entrada de conexiones TCP por el puerto 10115.
- Permitir tráfico entrante UDP por el puerto 10115 (sincronización de los relojes)
- Permitir tráfico entrante UDP (Puertos aleatorios)

De esta forma se asegura la conectividad entre los componentes de la aplicación.

En la consola de Chariot también debe cambiarse la configuración: En el campo dirección IP del E1 debe colocarse la palabra *localhost*. De esta forma, el E2 sabe que debe comunicarse con la dirección pública que se le asigna a E1 en el proceso de NAT, ya que si utilizara la dirección privada, sería imposible establecer la comunicación.

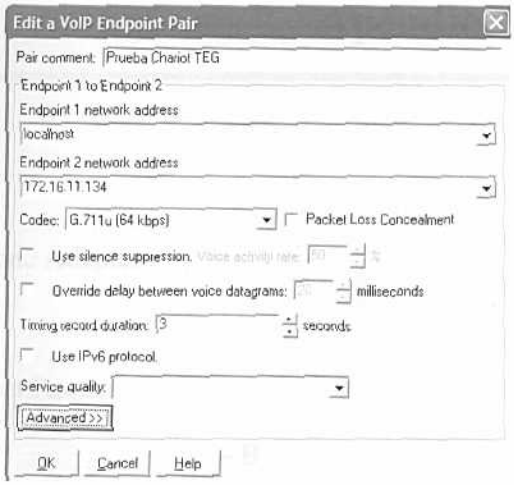


Figura 17: Configuración del par, bajo entornos NAT. (Autor)

De igual forma, el campo donde se especifica la dirección IP administrativa para el E1 debe desmarcarse, ya que sino la consola tomará los valores por defecto de la configuración del par, es decir la palabra *localhost*, lo que sería incorrecto:

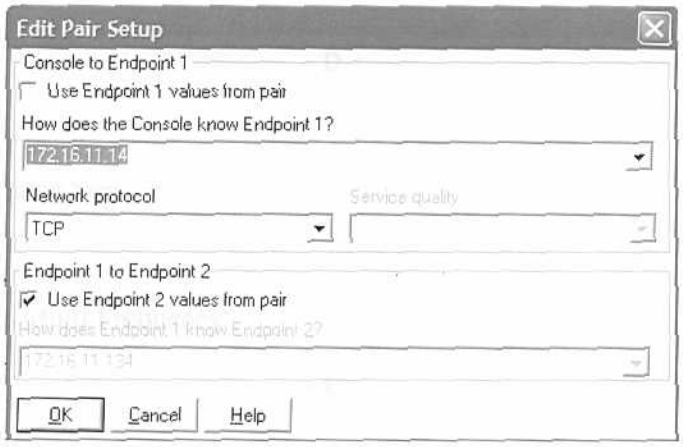


Figura 18: Configuración del par, bajo entornos NAT. (Autor)

Luego puede iniciarse la prueba sin problemas, y si éstos se presentan, debe asegurarse primero: la correcta configuración del equipo que realiza el NAT donde probablemente se genere el error, de lo contrario verificar la configuración del par.

- A -

ATA

Analog Telephone Adapter

ARR

Automatic Repeat Request

- B -

BER

Bit Error Rate

- C -

CN

Comfort Noise

- D -

DNS

Domain Name Server

DTMF

Dual Tone Multi Frequency

- E -

EHF

Extremely High Frequency

- F -

FEC

Forward Error Correction

- G -

- H -

HTTP

HyperText Transfer Protocol

- I -

IDU

Indoor Unit

IETF

Internet Engineering Task Force

IF

Intermediate Frequency

IP

Internet Protocol

ITU

International Telecommunications Union

ISP

Internet Service Provider

- L -

LAN

Local Area Network

- M -

MCPC

Multiple Channel Per Carrier

MEO

Medium Earth Orbit

MOS

Mean Opinion Score

- N -

NMS

Network Management System

- O -

ODU

Outdoor Unit

- P -

PEP

Performance Enhanceing Proxies

PESQ

Perceptual Evaluation of Speech Quality

PLC

Packet Loss Concealment

PSTN

Public Switched Telephone Network

PVC

Private Virtual Circuit

- Q -

QoS

Quality of Service

- R -

RF

Radio Frequency

RSVP

Resources Reservation Protocol

RTCP

Real-time Transport Control Protocol

RTP

Real-time Transport Protocol

RTT

Round Trip Time

RTSP

Real Time Streaming Protocol

- S -

SAP

Session Announcement Protocol

SCPC

Single Channel Per Carrier

SDP

Session Description Protocol

SHF

Super High Frequency

SIP

Session Initiation Protocol

- T -

TCP

Transport Control Protocol

- U -

UAC

User Agent Client

UAS

User Agent Server

UDP

User Datagram Protocol

UHF

Ultra High Frequency

UIT

Unión Internacional de Telecomunicaciones

URL

Universal Research Locator

- V -

VAD

Voice Activity Detection

VHF

Very High Frequency

VoIP

Voice over Internet Protocol

VSAT

Very Small Aperture Terminal

- W -

WAN

Wide Area Network

Wi-Fi

Wireless Fidelity

RTP Real-time Transport Protocol

Traducido al castellano como Protocolo de Transporte en Tiempo real, es un protocolo que funciona a nivel de la capa de aplicación del modelo OSI, el cual fue diseñado especialmente para transportar datos sensibles al retado como voz y video por el grupo de trabajo del IETF encargado de voz y video. Su primera estandarización se tradujo en el RFC 1889, el cual fue reemplazado posteriormente por el RFC 3550 en el 2003.

El objetivo principal de RTP es facilitar la entrega, monitoreo, reconstrucción, y la sincronización de los flujos de datos multimedia, aunque es importante aclarar que RTP no provee capacidades de calidad de servicio sobre redes IP como tal. RTP está diseñado para transmisiones *multicast* y *unicast* y no posee unos puertos predefinidos para TCP o UDP. En este sentido, el Standard si establece que para comunicaciones sobre UDP el puerto utilizado debe ser un número par, y de esta forma, el siguiente número de puerto impar es utilizado para comunicaciones sobre RTCP (*Real-time Transport Control Protocol*). Aun así, el protocolo por lo general está configurado para trabajar en un rango de puertos que abarca 16384-32767.

Realmente la principal ventaja de RTP, y la razón principal de su amplia aceptación es que básicamente agrega funcionalidades que UDP es incapaz de ofrecer, haciendo posible la transmisión de datos sensibles al retardo; pero disminuyendo el *overhead* que se necesitaría para llevar a cabo la misma transmisión utilizando TCP. Entre estas funcionalidades se encuentran, entre otras, número de secuencia y *timestamps* o marcas de tiempo. La estructura de un paquete RTP se muestra a continuación:

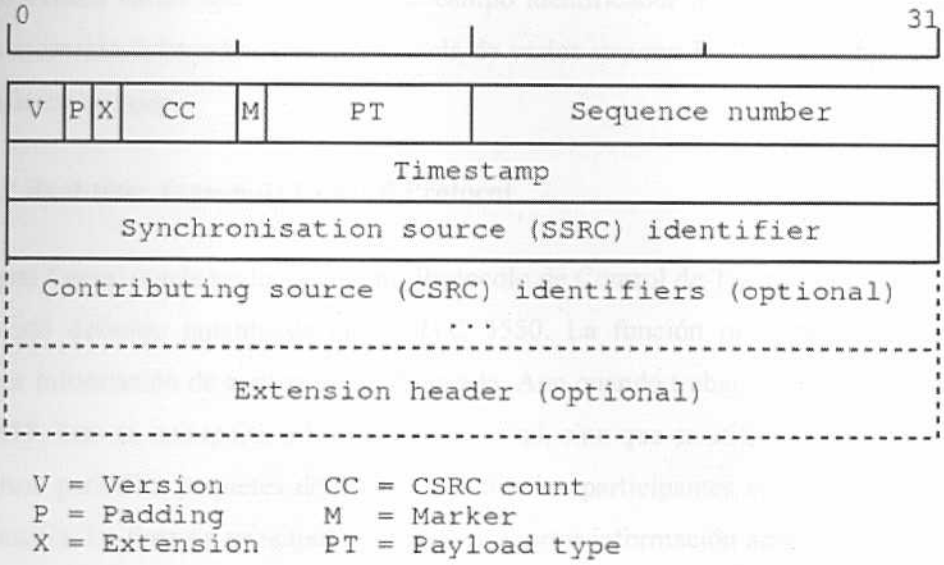


Figura 1. Estructura del paquete RTP. Fuente: Liesenborg, Jori. (http://research.edm.luc.ac.be/jori/thesis/onlinethesis/chapter5.html#5.2.2)

Los primeros dos bits de la cabecera RTP corresponden a la versión, y la versión actual del protocolo es la número dos. Seguidamente está el bit de relleno. El bit de longitud, o extensión indica si el paquete contiene una extensión del header. El campo CSRC especifica cuantas fuentes contribuyen a conformar el header RTP.

Seguidamente, y después del bit de marca, se encuentra un identificador del tipo de *payload* o carga útil, el cual define el tipo de data que se lleva dentro del *payload*, y define la forma en la cual una aplicación debe interpretar el *payload*. El número de secuencia puede ser utilizado por una aplicación para verificar que los paquetes sean recibidos en el orden correcto. Este número de secuencia al inicio de la comunicación es un número aleatorio.

La marca de tiempo o *timestamp*, contiene la información de sincronización para un flujo de datos. Este valor especifica cuándo el primer *byte* del *payload* fue muestreado. El campo SSRC o identificar de sincronización origen, es el número de identificación del origen del paquete, y permite establecer varias sesiones RTP por ejemplo para sesiones simultáneas de voz y video.

Luego existen varias opciones, como el campo identificador de origen o CSRC. Si en algún momento debe originarse una mezcla de varios *streams* RTP, este campo permite identificar el origen.

RTCP Real-time Transport Control Protocol

De igual forma, puede traducirse como Protocolo de Control de Transmisión en Tiempo real. Está definido igualmente en el RFC 3550. La función principal de RTCP es proveer información de control fuera de banda. Aun cuando trabaja de manera conjunta con RTP, éste no transporta información como tal, sino que es utilizado para transmitir de forma periódica paquetes de control a todos los participantes en un flujo de datos multimedia. La función principal de RTCP es aportar información acerca de la "calidad del servicio" prestada por RTP.

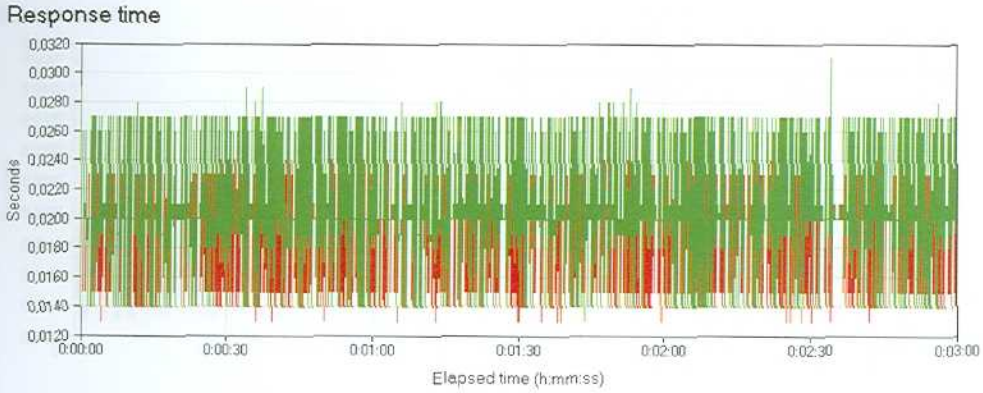
RTCP es capaz de recolectar estadísticas acerca de la conexión, como cantidad de *bytes* enviados, paquetes enviados, pérdida de paquetes, *jitter*, y RTT. Esta información puede ser utilizada por una aplicación, como por ejemplo la herramienta NetIQ Chariot, para incrementar de alguna forma la calidad del servicio, bien sea limitando el flujo de datos, o utilizando un codec que consuma más o menos ancho de banda, dependiendo de las características propias de la arquitectura de red.

Apéndice D Gráficas de Resultados

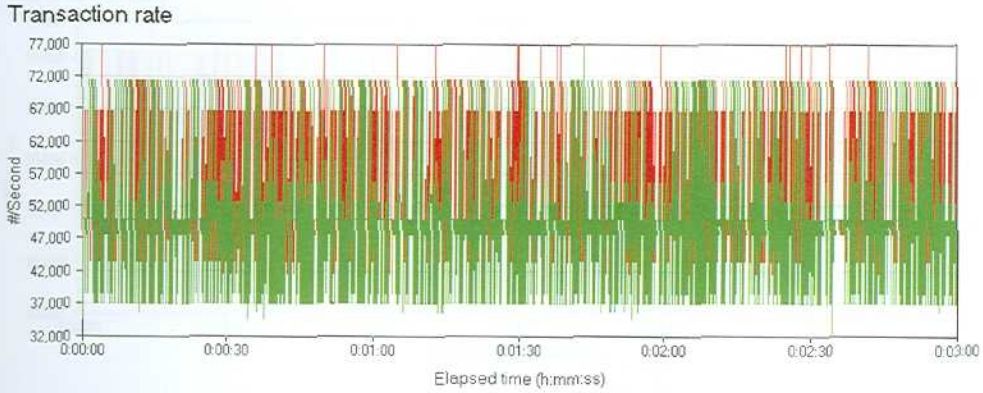
1 Realización de experiencias prácticas en las instalaciones del CAI.

En esta sección se presentaran las gráficas de los resultados que sustentan los análisis realizados.

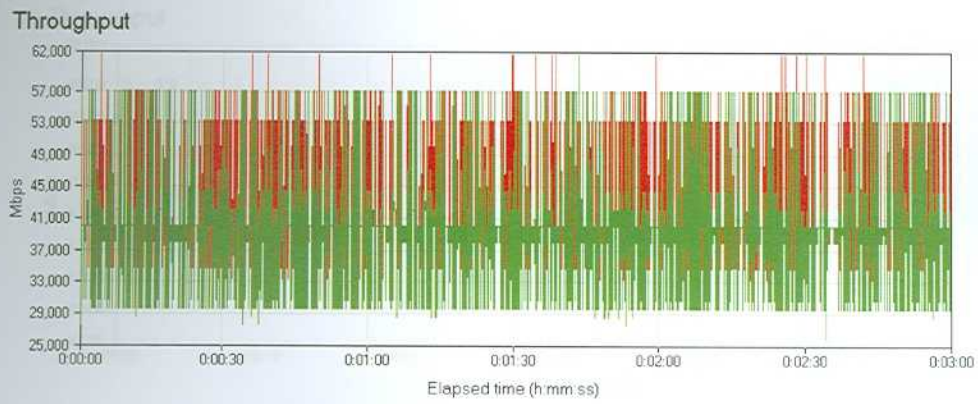
1.1 Caracterización del enlace



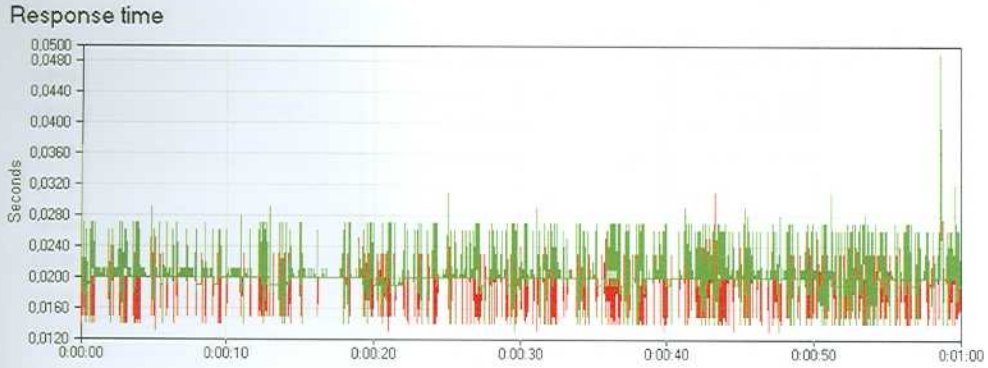
Gráfica 1. ***Response Time*** **en el sentido de extremo A extremo B. (Autor)**



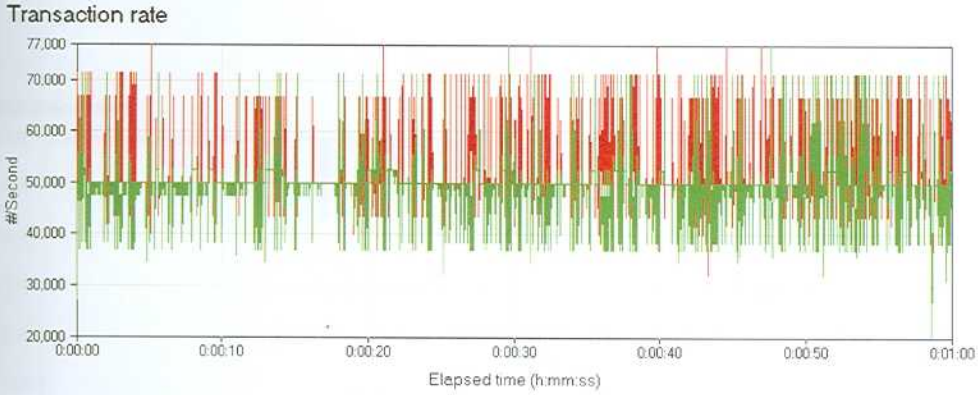
Gráfica 2 ***Transaction rate*** **en el sentido de extremo A extremo B. (Autor)**



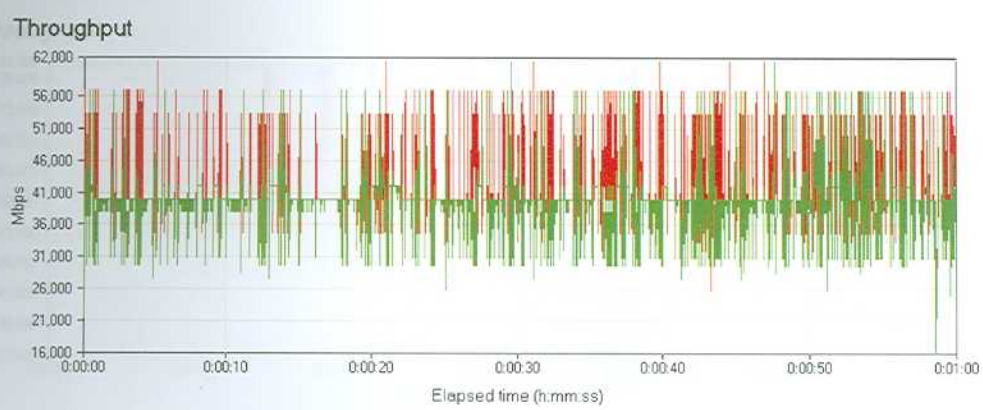
Gráfica 3. ***Throughput*** **en el sentido de extremo A a extremo B. (Autor)**



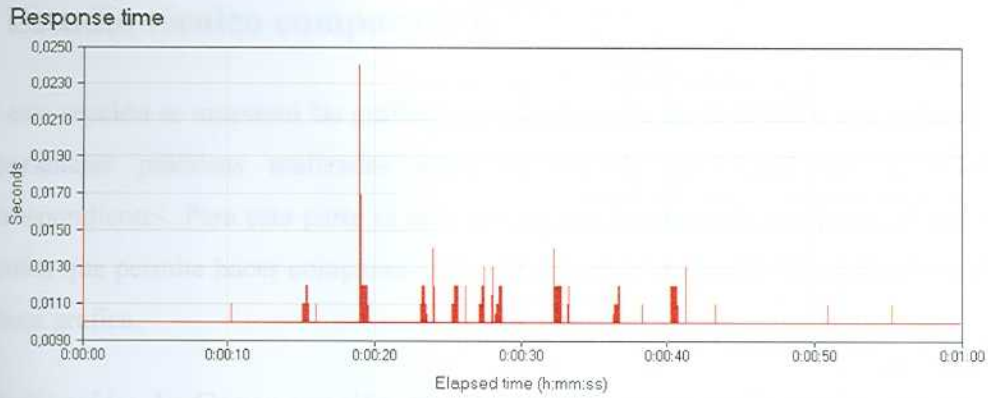
Gráfica 4. ***Response Time*** **en el sentido de extremo A extremo B. (Autor)**



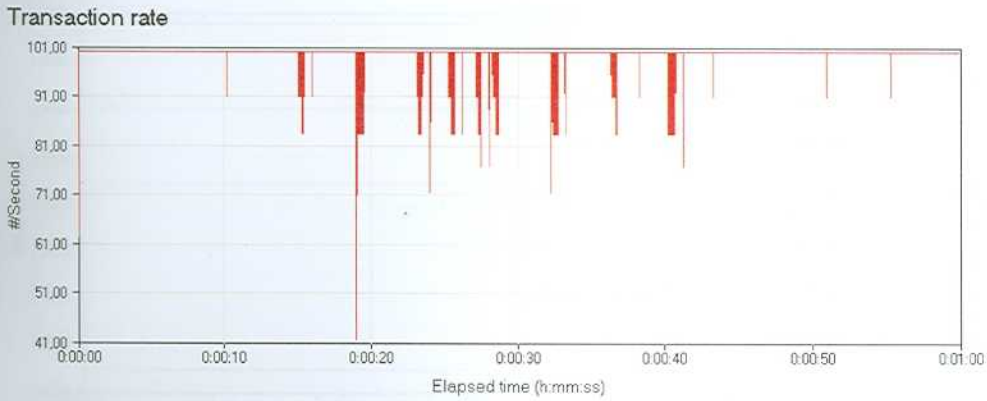
Gráfica 5. ***Transaction rate*** **en el sentido de extremo A extremo B. (Autor)**



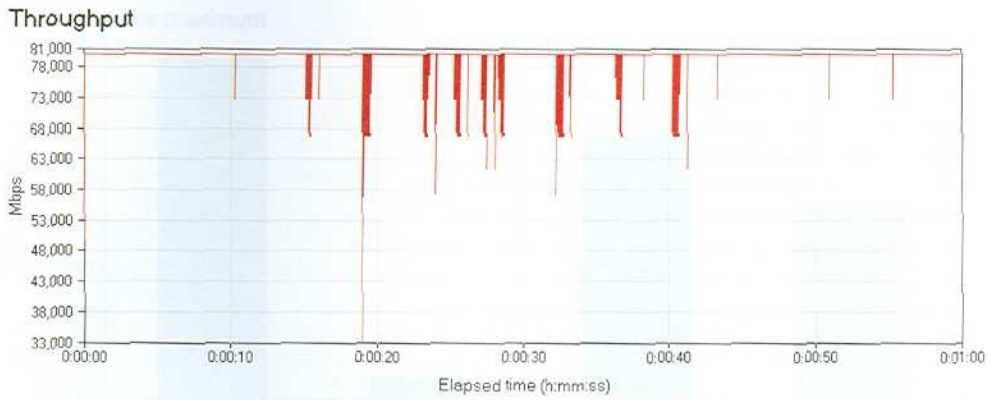
Gráfica 6. ***Throughput*** **en el sentido de extremo A a extremo B. (Autor)**



Gráfica 7. ***Response Time*** **en un solo sentido A----B. (Autor)**



Gráfica 8. ***Transaction rate*** **en un solo sentido A----B. (Autor)**

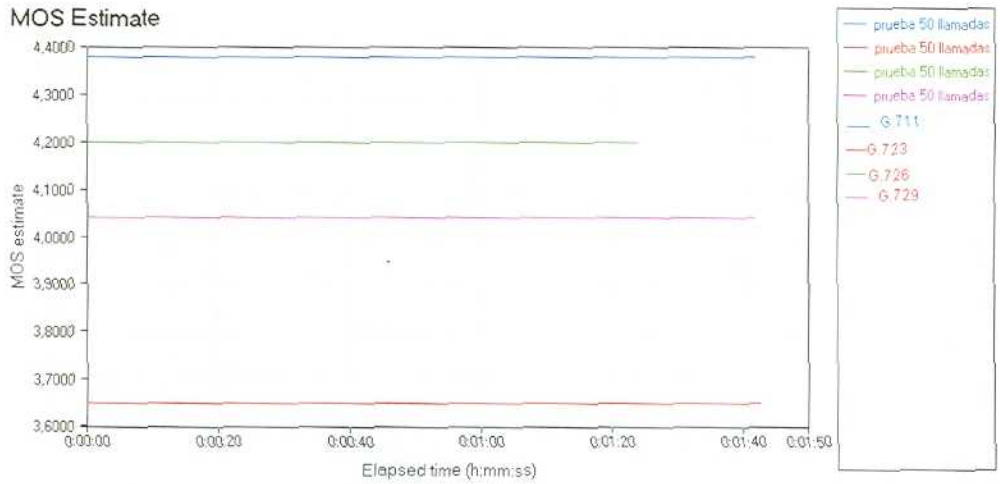


Gráfica 9. *Throughput* en un solo sentido A----B. (Autor)

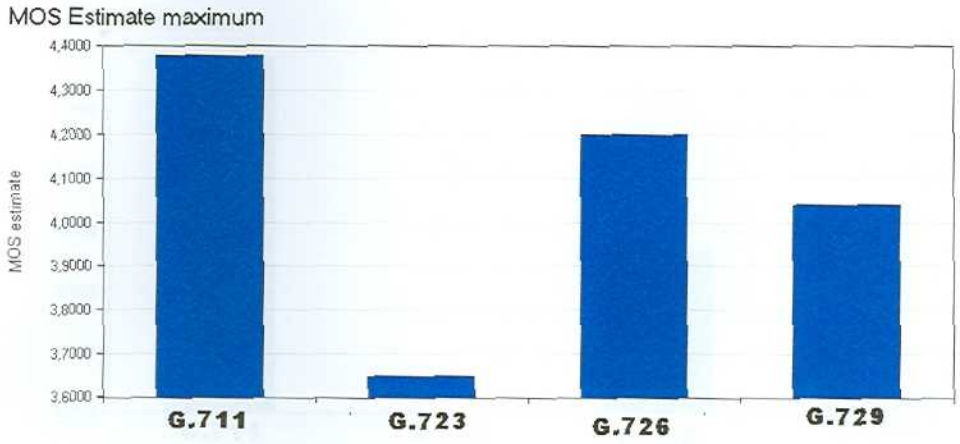
2 Estudio técnico comparativo

En esta sección se muestran las gráficas comparativas de los resultados obtenidos de las experiencias prácticas realizadas sobre la red de la UCAB en las VLANs correspondientes. Para esta parte se hizo uso de una herramienta que posee el *software* Chariot que permite hacer comparaciones entre pruebas arrojando los resultados en una misma grafica.

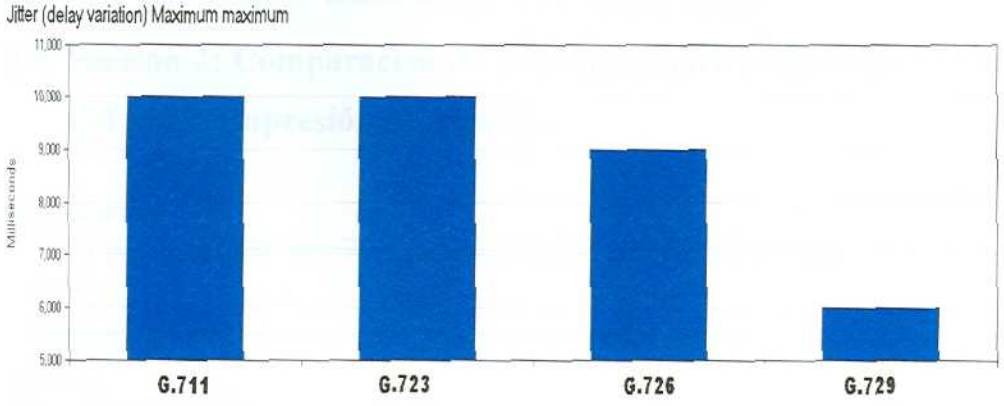
2.1 Sección 1: Comparación de resultados para cada codec sin el uso de supresión de silencio.



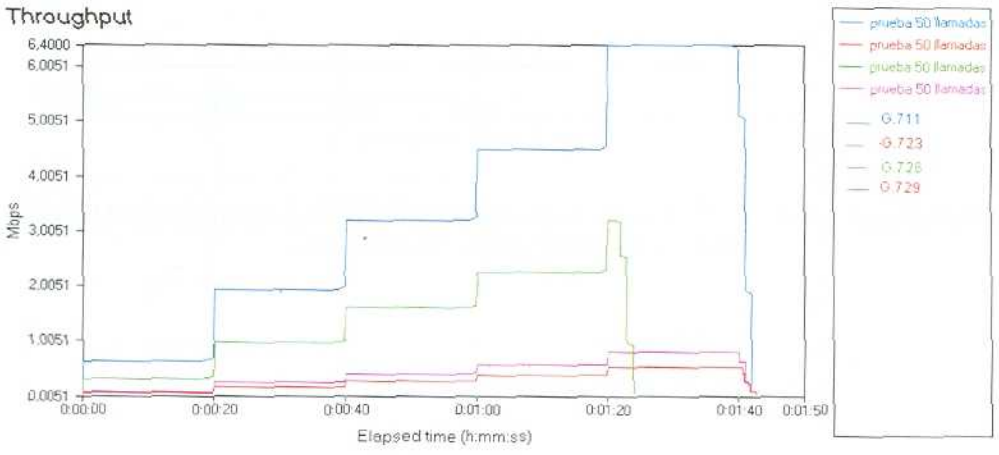
Gráfica 10. Comparación entre el MOS resultante en ambos sentidos de la conexión, sin supresión de silencio. (Autor)



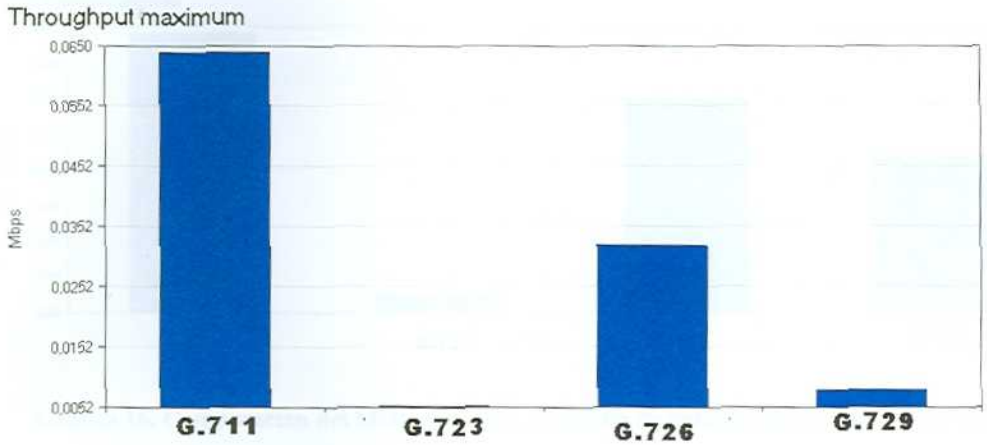
Gráfica 11. Comparación entre el MOS sin supresión de silencio. (Autor)



Gráfica 12. Comparación entre el jitter de los distintos codecs, sin supresión de silencio. (Autor)

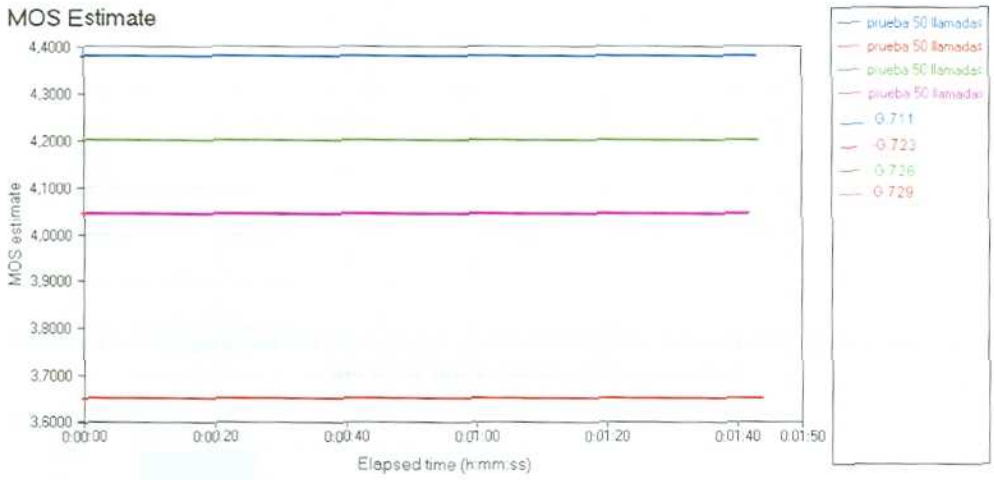


Gráfica 13. Comparación del *Throughput* en ambos sentidos de la conexión para los distintos codecs, sin supresión de silencio. (Autor)

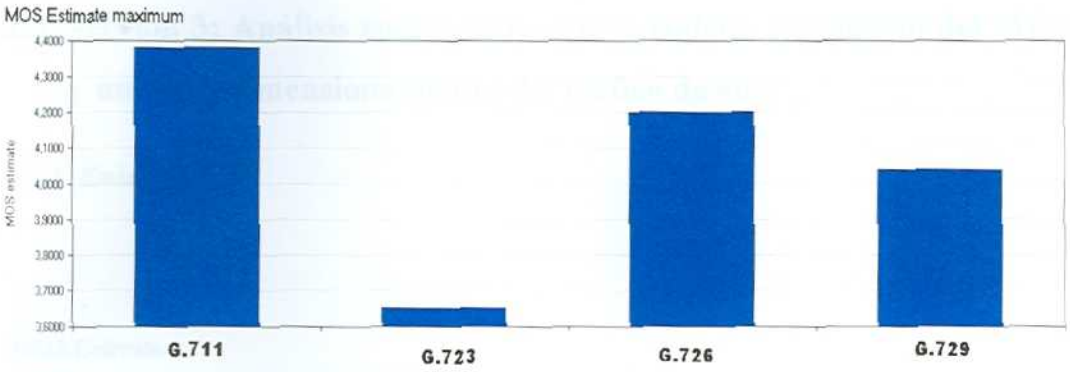


Gráfica 14. Comparación del *Throughput* en ambos sentidos de la conexión para los distintos codecs, sin supresión de silencio. (Autor)

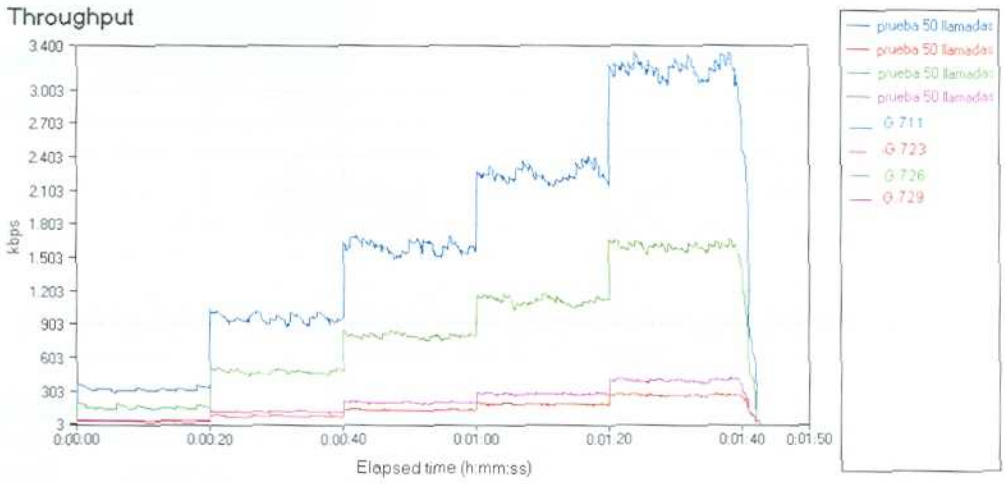
2.2 Sección 2: Comparación de resultados para cada codec en función del uso de supresión de silencio.



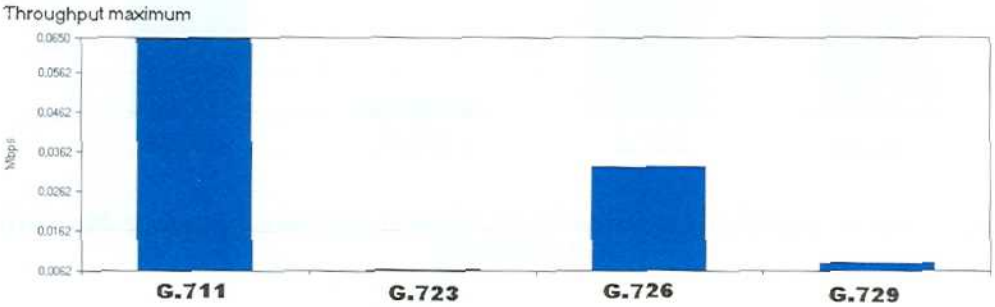
Gráfica 15. Comparación del MOS estimado, promedio de todas las llamadas, para todos los codecs con el uso de supresión de silencio. (Autor)



Gráfica 16. Comparación del MOS estimado, promedio de todas las llamadas, para todos los codecs con el uso de la supresión de silencio (Autor).



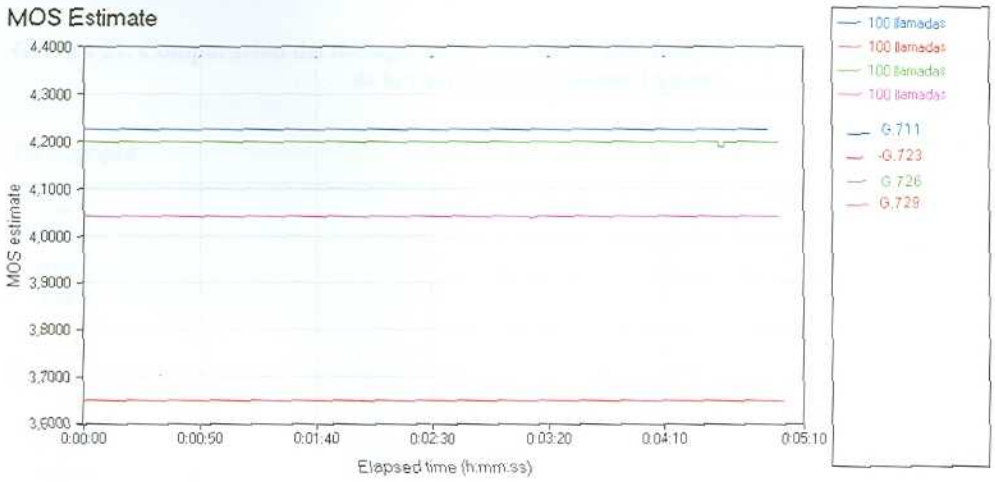
Gráfica 17. Comparación del *throughput*, promedio de todas las llamadas, para todos codecs con el uso de la supresión de silencio. (Autor)



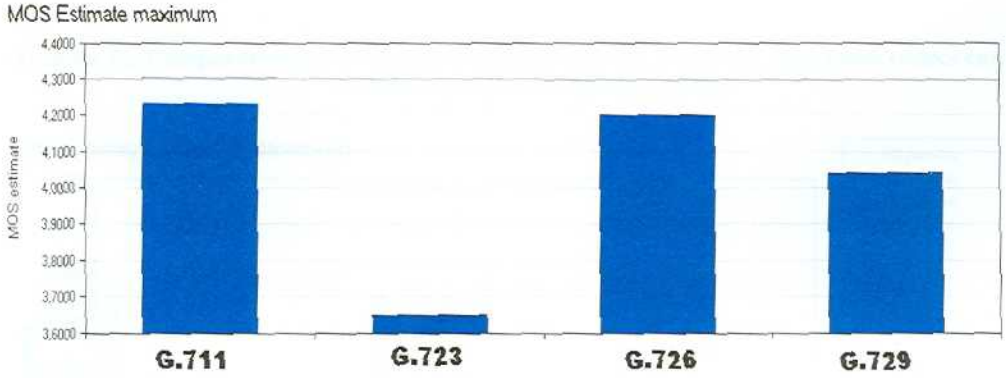
Gráfica 18. Comparación del *throughput*, promedio de todas las llamadas, para todos codecs con el uso de la supresión de silencio. (Autor)

2.3 Sección 3: Análisis comparativo entre codecs, en función del MOS y un sobredimensionamiento del tráfico de voz.

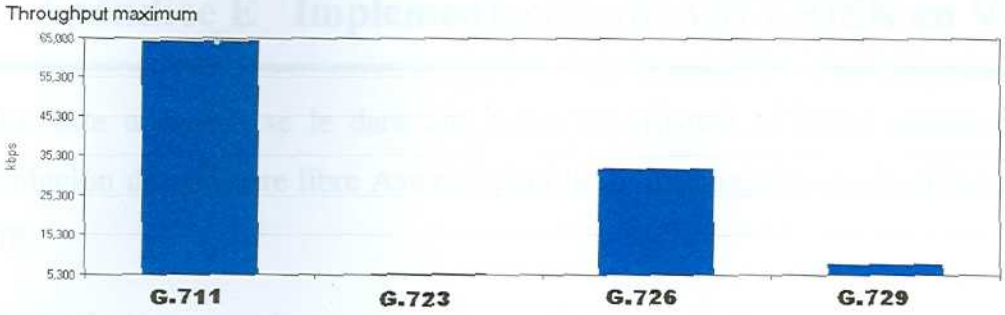
2.3.1 Enlace VLAN



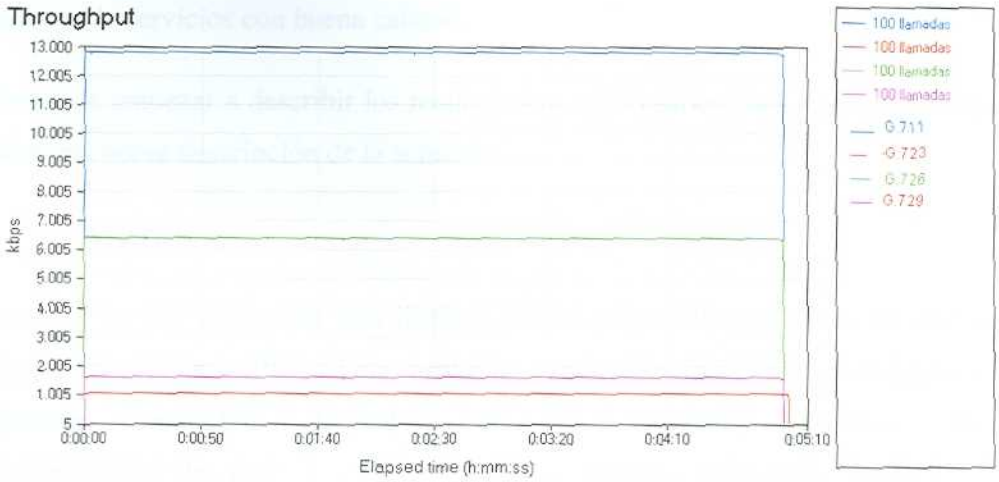
Gráfica 19. Comparación del MOS promedio, de 100 llamadas, para todos los codecs. Con supresión de silencio. (Autor)



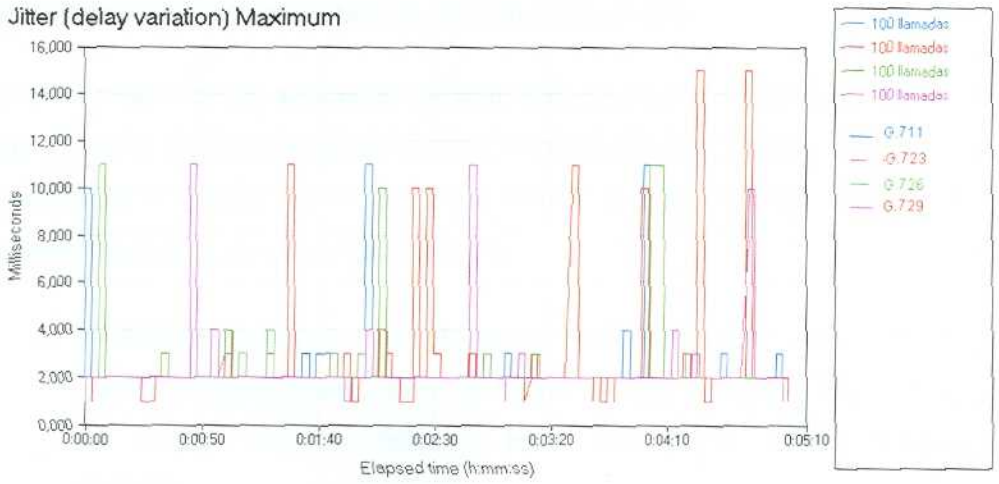
Gráfica 20. Comparación del MOS promedio, de 100 llamadas, para todos los codecs, con supresión de silencio. (Autor)



Gráfica 21. Comparación del *throughput*, promedio de 100 llamadas, para todos codecs con el uso de la supresión de silencio. (Autor)



Gráfica 22. Comparación del *throughput*, promedio de 100 llamadas, para todos codecs con el uso de la supresión de silencio. (Autor)



Gráfica 23. Comparación entre el jitter de los distintos codecs, con supresión de silencio. (Autor)

Apéndice E Implementación de ASTERISK en VoIP

En este apéndice se le dará una breve descripción al lector acerca de la solución de software libre Asterisk para la implementación de VoIP en redes IP.

Esta solución no propietaria Asterisk es una de las más implementadas por pequeñas y medianas empresas para interconectar sus redes telefónicas a través de VoIP, esto se debe a que es una de las soluciones mas rentables económicamente, y ofrece una amplia gamma de servicios con buena calidad.

Antes de empezar a describir los requerimientos necesarios para su implementación se hará una breve descripción de la solución.

1. **Asterisk**

Asterisk es una aplicación con licencia GPL (código abierto) capaz de simular las funciones de una IPBX. Este proyecto fue desarrollado por el ingeniero *Mark Spencer*, miembro fundador de la compañía *Digium*, principal desarrolladora de *Asterisk*. Originalmente fue implementado para cualquiera de las diferentes distribuciones *Linux* existentes, y aunque actualmente se está intentar portar a otros sistemas operativos como Mac, Solaris o Microsoft Windows, las expectativas son bastante pobres ya que carece de soporte.

Asterisk pese a ser una aplicación software ofrece las mismas características y servicios que los caros sistemas propietarios PBX como puede ser el buzón de voz, salas de conferencia o música en espera entre otros. Son muchas las ventajas que ofrece *Asterisk* respecto a las centralitas hardware:

- Reducción de costes y no sólo por el hecho de integrar voz y datos bajo una misma infraestructura, sino el hecho de que *Asterisk* sea una aplicación de código abierto evitando tener que pagar grandes cantidades por licencias.

- Facilita la integración y desarrollo de nuevos servicios de valor añadido.

- Compatibilidad con un gran número de protocolos VoIP y códecs.

- Es posible conectar *Asterisk* con otras centralitas, lo que le convierte en una solución flexible para futuros redimensionamientos.

- Existe un gran número de empresas y comunidades interesadas en el desarrollo de *Asterisk* que generan nuevas actualizaciones periódicamente. Debido a este gran interés existe una gran cantidad de información disponible.

Asterisk está formado por un núcleo principal encargado de gestionar todo el sistema PBX. Sus funcionales principales son:

- Interconectar de forma automática cada llamada ente los usuarios participantes teniendo en cuenta el tipo de protocolo utilizado por cada terminal.

- Lanzar los servicios de valor añadido cuando sean requeridos.

- Traducir y adaptar los códecs a cada terminal involucrado en la comunicación.

- Gestionar el sistema para que funcione de la forma más óptima en todas las condiciones de carga.

Para realizar estas funciones, este núcleo se apoya de un conjunto de módulos que le dotan de una gran flexibilidad y de una total abstracción de los protocolos, códecs e interfaces utilizados en cada conexión. Destacan cuatro APIs (*Application Programming Interface*) utilizadas por el núcleo de *Asterisk*

- **Channel API:** encargado de gestionar y extraer la información dinámica(protocolos, interfaces y códecs) de cada conexión.

- **Application API:** contiene diferentes módulos encargados de ofrecer distintos servicios de valor añadido. Esta estructura modular facilita la incorporación de nuevos servicios.

- **Codec Translator API:** permite cargar los diferentes formatos de códecs de audio utilizados para la compresión y codificación de la señal. Al

igual que las aplicaciones, estos códecs están implementados como módulos independientes.

- **File Format API:** permite leer y escribir ficheros para el almacenamiento de información en el sistema de archivos, como por ejemplo, la grabación de una conversación.

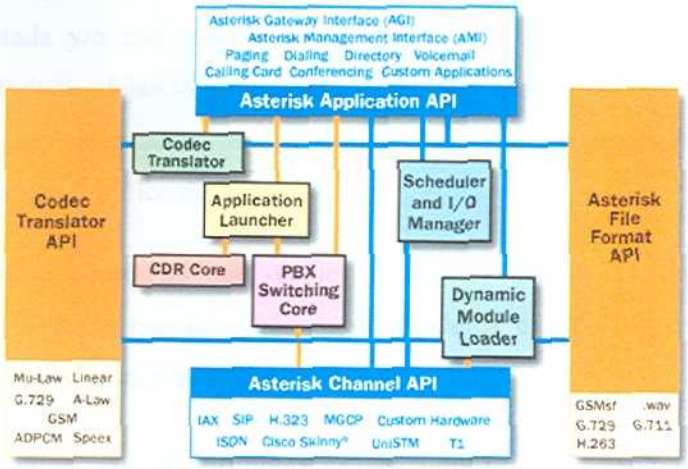


Fig. 1.2 Núcleo principal de *Asterisk*

Tal y como se puede observar en la Tabla 1.1, otra de las características que hace de *Asterisk* una solución interesante es el soporte de varios protocolos VoIP y códecs de audio.

Protocolos VoIP	SIP	H.323	IAX	UNISTIM[2]	MGCP/MEGACO	SCCP1	
Códecs de audio	G.711(a-law y µ-law)	G.726	G.723.1	G.729A	GSM	iLBC	Speex ADPCM Linear MP3

Tabla 1.1 Protocolos VoIP y códecs de audio soportados por *Asterisk*.

Asterisk puede actuar de servidor y gateway. Para conectar esta IPBX a la red PSTN, a la RDSI o bien conectarle teléfonos analógicos no basta con un simple módem sino que son necesarias unas tarjetas telefónicas con puertos FXO, BRI/PRI o FXS respectivamente. La comunicación entre el servidor y las tarjetas se efectúa a

través de otro módulo externo, llamado *zaptel*, que actúa como controlador de la tarjeta. Además si el acceso al exterior se realiza mediante puertos primarios de una red RDSI se necesita un segundo módulo complementario llamado *libpri*.

2. **Tarjetas analógicas FXO/FXS**

Las tarjetas analógicas permiten conectar el servidor *Asterisk* con la red de telefonía conmutada y/o con teléfonos analógicos. Uno de los principales ingresos de la empresa *Digium* , dejando al margen el soporte y mantenimiento de servidores *Asterisk*, es la implementación, distribución y venta de tarjetas PCI compatibles dedicadas a gestionar el tráfico de voz.

TDM es el nombre comercial que recibe las tarjetas analógicas desarrollados por *Digium* y está formada por un conjunto de módulos extraíbles que permiten la conversión analógico-digital y viceversa:

- FXO (módulos rojos): permite conectar tantas líneas externas de la PSTN como módulos.
- FXS (módulos verdes): permite conectar tantos teléfonos analógicos como módulos.

Existen básicamente dos familias de tarjetas analógicas:

- **TDM400P:** tarjeta mini-PCI 2.2 con capacidad de conectar hasta 4 interfaces FXO (nomenclatura comercial: X100M), 4 módulos FXS (S110M) o cualquier combinación de estos con un máximo de 4 módulos. *Digium* ha asignado a este grupo de tarjetas la nomenclatura TDM X Y B, siendo X el número de interfaces FXS que incorpora la tarjeta, e Y el número de interfaces FXO. Por tanto, la tarjeta TDM13B (ver Fig. 2.1) es una tarjeta de la familia TDM400P con un módulo FXS y tres interfaces FXO.

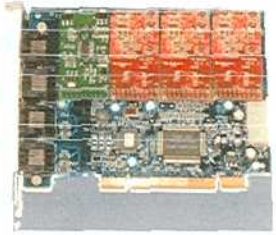


Fig. 2.1 TDM13B

- **TDM2400P:** tarjeta PCI 2.2 con capacidad de conectar hasta 6 módulos X400M (4 interfaces FXO), 6 módulos S400M (4 interfaces FXS) o cualquier combinación de estos con un máximo de 24 interfaces. Por tanto, la tarjeta TDM2451B (ver Fig. 2.2) es una tarjeta de la familia TDM2400P con cinco módulo X400M y un módulo S400M, equivalente a tener 20 interfaces FXO y 4 FXS.

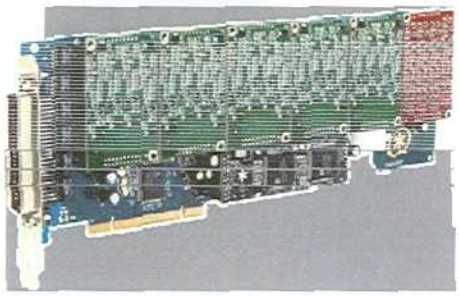


Fig. 2.2 TDM2451B

El precio de cada una de las tarjetas dependerá del número de módulos necesarios. En el caso de prever un futuro crecimiento de la red, la opción más rentable es sobredimensionar en cierta medida el sistema; ya que el precio de adquirir posteriormente los módulos es mayor que comprar inicialmente una tarjeta con un mayor número de puertos. Paralelamente, otras marcas como *Sangoma*, *Junghanns*, *Rhino* o *Eicon Networks*, entre otras, sacan tarjetas clónicas con características similares con un menor coste.

2.2 Tarjetas digitales

La función de estas tarjetas es similar a la de las analógicas, pero en este caso estas interfaces digitales permiten conectar *Asterisk* con una red digital RDSI. De este tipo de tarjetas se pueden encontrar dos tipos:

- **Básicas (BRI):** cada puerto BRI permite mantener 2 conversaciones simultáneamente. *Digium* no fabrica este tipo de tarjetas ya que en América las redes RDSI básicas no se han extendido. Para conseguir este tipo de tarjetas hay que recurrir a alguna de las compañías nombradas en el anteriormente. Estas empresas ofrecen tarjetas de:

9 **1 puerto** = 2 conversaciones simultáneas.

9 **4 puertos** = 8 conversaciones simultáneas.

9 **8 puertos** = 16 conversaciones simultáneas.

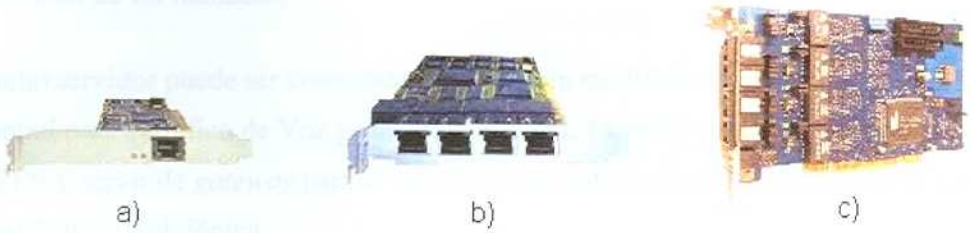


Fig. 2..3 Tarjetas BRI. a) *Eicon Networks Diva Server V-BRI*, b) *Eicon Networks Diva Server 4V-BRI*, c) *OctoBRI Junghams*

- **Primarios (PRI):** cada primario (E1/T1) permite establecer aproximadamente 30 conversaciones simultáneas. *Digium* comercializa varias tarjetas digitales con diferente número de puertos primarios:

- **primario**= 30 conversaciones simultáneas.

- **primarios**= 60 conversaciones simultáneas.

- **primarios**= 120 conversaciones simultáneas.

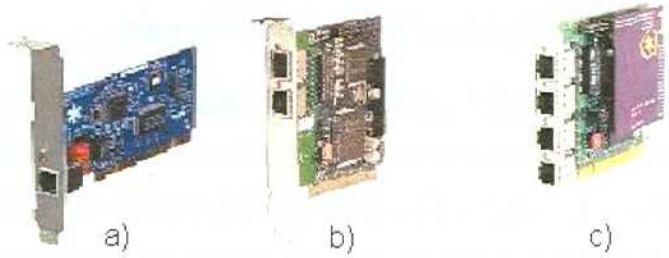


Fig. 2.4 Tarjetas PRI. *Digium* a) *TE110P*, b) *TE210P*, c) *TE412P*

3. Esquema de funcionamiento

El funcionamiento de *Asterisk* es bastante sencillo ya que un solo servidor es el encargado de funcionar como PBX y desarrollar todo el procesamiento, señalización y conexión de las llamadas.

Dicho servidor puede ser conectado a la LAN sin modificar su estructura y servir de central para el trafico de Voz generado por la red. También puede ser conectado a la PSTN y servir de *gateway* para la interconexión entre la Red Privada Telefónica y la Red Publica Telefónica.

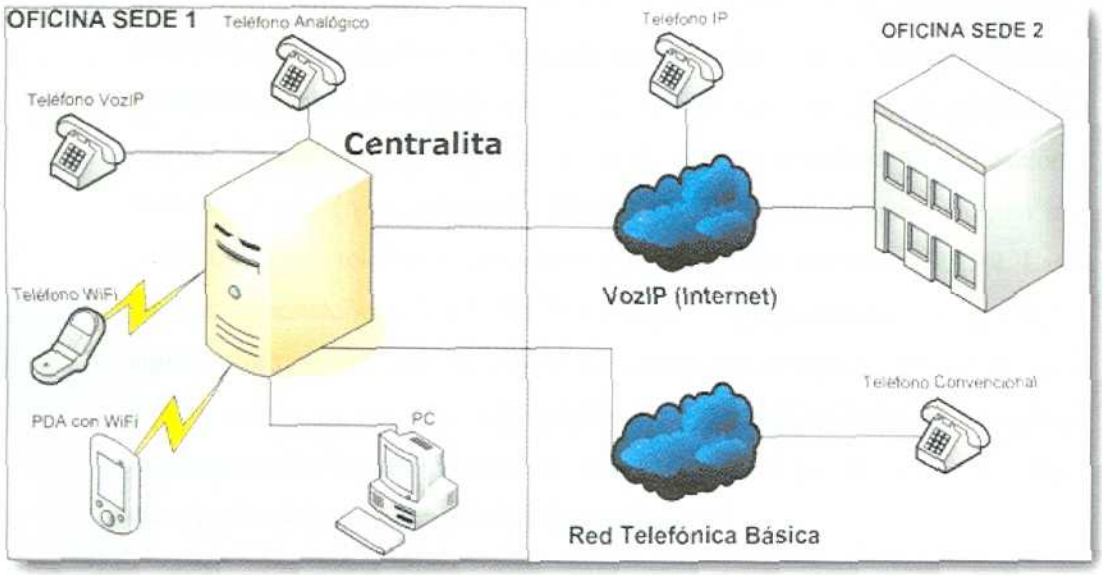


Figura 2.5. Descripción de la red Asterisk.

El funcionamiento es muy sencillo, el servidor se conecta a la red LAN en el cual se esta corriendo la aplicación Asterisk, y este a su ves mediante la conexión de tarjetas puede conectarse a los distintos tipos de dispositivos telefónicos dimensionados en la red. También se establecen conexiones con la PSTN permitiendo la conexión de llamadas que provengan del exterior de la red interna.

A través de protocolos de sesión y señalización como SIP, H.323, IAX, MEGACO entre otros, el servidor conjuntamente con la aplicación Asterisk puede establecerse como central PBX e interconectar los abonados previamente reconocidos por el sistema. Una ves que la central este conectada y los abonados sean reconocidos asignándoles una

dirección IP que seria el nuevo número telefónico que reconoce la central, mediante el protocolo de sesión que es el responsable del establecimiento de la comunicación se conecta la llamada a través de la red de datos hasta el destino final.

Es importante mencionar que depende del protocolo de sesión que se use en la implementación de la tecnología para asegurar el buen funcionamiento y la calidad de voz final.

4. Recomendaciones para la implementación de la Tecnología en las Redes de la UCAB.

- Para la implementación de VoIP en la red UCAB cede Antimano, se recomienda usar la solución Asterisk, ya que es la solución mas económica y mas adaptable a los requerimientos de la red. Esta solución necesitaría la compra de un equipo de servidor en el cual se instalaría la aplicación Asterisk. También se necesitaría las tarjetas de expansión que permitan la conexión de los teléfonos analógicos y la conexión entre la red PSTN. Las tarjetas de conexión de los teléfonos analógicos representaría un ahorro muy significativo ya que con su uso no se necesita la compra teléfonos IP los cuales son costosos. La administración y puesta en marcha de esta implementación puede dejarse en manos del CAI ya que ellos son los encargados del mantenimiento y soporte de la red.

- Recomendable implementar políticas de calidad de servicio, de forma tal de darle preferencia al tráfico de voz, dentro del núcleo de la red. Esto puede hacerse mediante el uso de protocolos de enrutamiento en los equipos de red que se encargan de marcar aquellos paquetes de voz para ofrecerles prioridad. Por ejemplo RSVP y MPLS.

- Es recomendable segmentar la red desde el punto de vista lógico, haciendo uso de VLAN y poder clasificar los servicios de la red de VoIP de acuerdo a los requerimientos de cada grupo de abonados.