ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"DESARROLLO DE SISTEMAS ABIERTOS DE SEÑALIZACIÓN SS7"

TESIS DE GRADO

PREVIO A LA OBTENCIÓN DEL TÍTULO DE:

MAGISTER EN TELECOMUNICACIONES

PRESENTADO POR:

José Miguel Menéndez Sánchez

GUAYAQUIL – ECUADOR

2013

AGRADECIMIENTO

Esta etapa académica que con gran entusiasmo y dedicación he cumplido, es el logro de más de una persona, las cuales juntas sumaron fuerzas para poder alcanzar la meta trazada. Por ello quisiera expresar mis agradecimientos a:

Dios por todas sus bendiciones recibidas.

Mis pequeños, José Elías y José Daniel, que comprendieron cuando les hice falta en sus juegos diarios.

A Gladys, mi esposa, que suplió mi rol como cabeza de familia y supo ser compañera de juegos de nuestros hijos en mis ausencias.

A Ángel, mi jefe, por su valiosa ayuda en la flexibilidad de los horarios para poder asistir a clases. Y a mis colegas, Héctor Paredes Leite de la Universidad Nacional de Asunción y a Dai Yonglong de la Universidad Beihang de Beijing por su guía y aporte en este trabajo.

DEDICATORIA

El tiempo que representa esta maestría, es el tiempo de mi familia, de mis amigos y de mi trabajo. Tiempo sacrificado por alcanzar una meta y subir un peldaño académica y profesionalmente. En su mayoría es el tiempo lejos de los juegos de mis hijos, de las conversaciones de mi esposa, de los paseos y crecimiento familiar, por ello este logro está dedicado a lo más importante de mi vida y mi fortaleza como ser humano: mis hijos, que gracias a su compresión y apoyo, pude cumplir con el objetivo trazado desde el primer día de clase.

Así mismo, el titulo se lo dedico a la memoria del Ing. Christian David Robayo (+), mi gran amigo y compañero de largas horas de estudios.

TRIBUNAL DE SUSTENTACION

MSc. Miguel Yapur

Decano FIEC

PhD. Álvaro Suárez

Director de Tesis

PhD. Boris Ramos

Vocal Principal

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesis de Grado, me corresponde exclusivamente; y el patrimonio intelectual de la misma a la Escuela Superior Politécnica del Litoral".

(Art. 12 del Reglamento de Graduación de la ESPOL)

ING. JOSE MIGUEL MENENDEZ SANCHEZ

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

GLOSARIO

ACM	Address complete message
ANSI	American National Standards Institute
ASP	Application Server Process
BICC	Bearer Independent Call Control
CAP	Camel Application Part
CAPEX	CAPital EXpenditures
CDI	Communications Device Interface
CH	Channel
CIC	Circuit identification code
CON	Connect Message
CS/AIN	Capability Set/Advanced Intelligent Network
DASS	Digital Access Signaling System
DLPI	Data Link Provider Interface
DPC	Destination Point Code
DPNSS	Digital Private Network Signalling System
ENUM	E.164 NUmber Mapping
FISU	Fill In Signal Unit
GCC	GNU Compiler Collection
GSM	Global System for Mobile Communication
GPL	General Public License

GPRS	General Packet Radio Service
GTT	Global Tittle Translation
H.248	MEGACO
HLR	Home Location Register
HTTP	Hyper Text Transfer
IAM	Initial Address Message
IAX	Inter-Asterisk eXchange
IN	Intelligent Network
INAP	Intelligent Network Application Part
IP	Internet Protocol
IPSEC	Internet Protocol Security
IPSP	IP Security Policy
IPSS7	IP SS7
ISA	Industry Standard Architecture
ISDN	Integrated Services Digital Network
ISUP	ISDN User Part
ITU-T	International Telecommunication Union - Telecommunication
LAPB	Link Access Procedure, Balanced
LAPD	Link Access Protocol for D-channel
LDL	Linux Data Link
LiS	Linux Stream

LKSCTP	Linux Kernel SCTP
LNP	Local Number Portability
LSSU	Link Status Signal Unit
M2PA	Message Transfer Part 2 User Peer-to-Peer Adaptation
M2UA	MTP Level 2 User Adaptation
M3UA	MTP Level 3 User Adaptation
MAP	Mobile Application Part
MEGACO	Media Gateway Control
MG	Media Gateway
MGC	Media Gateway Controller
MGCP	Media Gateway Control Protocol
MSU	Message Signal Unit
MTP-1	Message Transfer Part level 1
MTP-2	Message Transfer Part level 2
MTP-3	Message Transfer Part level 3
MX	Multiplex
NAPTR	Name Authority Pointer
NPI	Network Provider Interface
OPC	Originating Point Code
OPEX	OPerating EXpenditures
OSI	Open Systems Interconnection
PBX	Private Branch eXchange

PCI	Peripheral Component Interconnect
PSTN	Public Switched Telephone Network
REL	Release message
RLC	Release Call message
RPM	Red Hat Package Manager
RTO_INI	Retransmission Time-out Initial
RTO_MAX	Retransmission Time-out Maximal
RTO_MIN	Retransmission Time-out Minimum
SCN	Switching Circuit Network
SCP	Signaling Control Point
SCCP	Signaling Connection Control Part
SCTP	Stream Control Transmission Protocol
SG	Signaling Gateway
SGP	Signaling Gateway Process
SIGTRAN	Signaling Transport
SIP-T	Session Initiation Protocol for Telephones
SMPP	Short Message Peer-to-Peer
SMS	Short Message Service
SMSC	Short Message Service Center
SS7	Signaling System #7
SSN	Subsystem Number
SSP	Signaling Service Point

STP	Signal Transport Point
SUA	SCCP User Adaptation
SVR4.2	System V Release 4.2
TALI	Transport Adaptation Layer Interface
TCAP	Transaction Capabilities Application Part
TCP/IP	Transmission Control Protocol over IP
TDM	Time Division Multiplexing
TLS	Transport Layer Security
TPI	Transport Provider Interface
TUA	TCAP User Adaptation
IUA	ISDN User Adaptation
TUP	Telephone User Part
UDP	User Datagram Protocol
UN*X	Unix-like
VoIP	Voice over IP
WAP	Wireless Application Protocol
X.25	Protocolo X.25
XNS	X/Open Networking Services
XTI	X/Open Transport Interface

INTRODUCCIÓN

Sin lugar a duda, los sistemas de señalización son los que permiten el paso de tráfico tanto de voz y datos en las amplias redes de comunicaciones. Por lo que la señalización ha llegado a posicionarse como un tópico que no solo representa aspectos técnicos, sino inclusive ha llegado a tener el enfoque económico y rentable gracias a empresas dedicadas al desarrollo de protocolos propietarios, en su gran mayoría bajo el *Sistema de Señalización por Canal Común número 7* (*SS7*), el cual yergue como el más difundido para el paso principalmente de llamadas, así como traducción de números, conexión a la *Red Inteligente*, aplicaciones a nivel de *Parte de Aplicación Móvil (Mobile Application Part, MAP)*, entre otros.

Que los protocolos SS7 sean propietarios provoca un impacto muy considerable dentro de las compañías de telefónicas, puesto que la red no sólo pasa a depender de las licencias de uso de SS7 sino también de los equipos en donde se ejecutan las aplicaciones; a más de esto se debe agregar que los costos de mantenimiento preventivos como reactivos alcanzan valores ineludibles para las mencionadas empresas. Por lo tanto, el

triángulo Software-Hardware-Licencia es un aspecto con el que deben cargar las operadoras, y el tema de SS7 no es una excepción.

CAPÍTULO 1

1 ANTECEDENTES Y JUSTIFICACIÓN

1.1 Antecedentes de SS7 abierto, OpenSS7

Aunque en sus inicios el término de *Open Source*, o Código abierto en español, sirvió para eliminar la ambigüedad generada por su término predecesor: *Free Software*, debido al doble significado en inglés de gratis y libre, aun bajo ciertos contextos tiene referencias ambivalentes acerca del uso y la gratuidad.

Sin embargo, *Código Abierto*, es adecuado para referirse a programas que se ofrecen libremente para la modificación, uso y redistribución, con la condicionante de que este principio no cambie para las variantes de programas generados, es decir que *Código Abierto* no solo se refiere al

acceso del código fuente, sino también a los puntos previamente mencionados [1]

Dada la necesidad de no verse atado a un software propietario, y contar con la flexibilidad de un sistema operativo personalizable, Richard Stallman inicia en 1983 el proyecto GNU (acrónimo recursivo pronunciado ñu y que significa No Unix), el mismo que nace con el objetivo de crear un sistema operativo completamente libre. Sin embargo, este proyecto requería de un soporte legal, logístico y financiero, que lleva a Stallman a fundar en 1985, la Fundación del Software Libre [2]

De esta manera, el software libre que se publica debe tener una licencia de software libre, y una de las licencias más difundidas es la Licencia Pública General (*GPL* por siglas en inglés) GNU, cuyo objetivo primordial es hacer el papel de protector del software libre publicado por la Free Software Foundation, en cuanto a la distribución, uso y modificación del software cubierto por esta licencia, es decir la licencia concesiona ciertos derechos al usuario que la adquiere [3]

SS7 abierto o también llamado OpenSS7, es un proyecto de *Código Abierto* iniciado en 1996, por OpenSS7 Corporation, cuyo objetivo primordial es brindar de manera robusta la pila del protocolo SS7 y bajo licencia LPG para Linux y otros sistemas operativos *UN*X* (Linux, Mac OS X, Solaris...) [4]

1.2 Justificación del uso de SS7 abierto

El desarrollo de sistemas abiertos de SS7 pretende desvincular la ligación entre licencias de uso de protocolos a nivel de SS7, lo cual se vuelve esencial en la optimización de recursos económicos de las empresas de telefonía como lo son el *CAPital EXpenditures* (*CAPEX*) y *OPerating EXpenditures* (*OPEX*).

Este proyecto abre las puertas al desarrollo de aplicaciones a nivel de señalización, bajo los parámetros que rigen la estandarización de la ITU-T sobre SS7, bajo licencia LPG. Así mismo, en la parte académica, el desarrollo de protocolos de SS7 sobre sistemas abiertos permite despegar hacia un camino de conocimientos a los estudiantes de pregrado y post-grado, dando como resultado una formación integral.

1.3 Descripción del Proyecto

Entre las características del mundo tecnológico, existe una especial que ha sido prácticamente una marca de nacimiento: el sentido de pertenencia de una marca específica sobre cada una de las novedades y características que sus productos ofrecen. Este aspecto no ha sido indiferente con la telefonía, donde existen muchos detalles que deben ser especificados hasta al más mínimo para poder lograr bondades como la escalabilidad, compatibilidad e integración. Para cada una de estas especificaciones físicas, lógicas o de protocolos usados, existe una marca detrás que no solo responde por el

soporte requerido sino también que ha vendido a un buen precio esta concesión, bajo la figura de licencia, para el uso y más no para la modificación.

Sin embargo, no todos estos aspectos están desarrollados bajo la necesidad de una licencia, en la actualidad el uso de licencias libres es más común y hasta necesario en las empresas (por optimización de recursos económicos).

El presente trabajo, pretende mostrar a OpenSS7 como una alternativa para uso de señalización entre equipos de telefonía que lo requieran, con el mismo desempeño, rendimiento, seguridad y compatibilidad con que se manejan las soluciones propietarias.

La implementación del servidor OpenSS7 consta de tres partes:

Instalación de los paquetes del software

En esta fase, es necesario contar con un servidor con sistema operativo recomendado por OpenSS7 Corp., el escogido para el desarrollo del presente trabajo es Fedora Core 9 con kernel 2.6, el mismo que actúa como anfitrión y es la base para soportar los paquetes y componentes de software OpenSS7. Sin embargo, la instalación de los mencionados componentes implica la preparación del servidor, como lo es el aplicar las dependencias necesarias para ejecutar cada instalación.

Configuración de componentes de Señalización

En una fase posterior, se realiza la configuración necesaria de los archivos de los componentes ya instalados que permiten integrar el servidor SS7 a la red telefónica, como por ejemplo a un *Punto de Transferencia de Señal (Signal Transfer Point, STP)*.

Pruebas de los Componentes Instalados

En esta instancia, se hace uso de los *scripts* que también son descargados durante la instalación de cada uno de los paquetes. Los *scripts* están orientados a probar la correcta instalación y operación del componente de señalización instalado.

Para estas etapas, se requiere de conocimientos profundos tanto de Linux, SS7 y *Transporte de Señalización (Signaling Transport, SIGTRAN)*, así como de los diferentes protocolos que usan el canal común para su operación: *Parte de Usuario para la Red de Distribución de Servicios Integrados* (*ISDN User Part, ISUP*), *Parte de Aplicación Camel* (*Camel Application Part, CAP*), *Parte de Aplicación de Red Inteligente* (*Intelligent Network Application Part, INAP)*, *Parte de Control de Conexión de Señalización (Signaling Connection Control Part*, *SCCP*), entre otros.

1.3.1 Objetivo general

El desarrollo de este trabajo tiene como objetivo primordial: proveer la documentación y sustentación suficiente para lograr la implementación de SS7 sobre sistemas operativos de *Código Abierto* como es el Linux.

1.3.2 Objetivos específicos

1. Detallar el alcance de SS7 en el mundo de la telefonía.
2. Mostrar las bondades del sistema operativo Linux para ser un buen anfitrión de los SS7 abiertos.
3. Exponer los avances realizados en la integración de SS7 sobre sistemas operativos abiertos.
4. Posicionar a los SS7 abiertos como alternativa a los SS7 propietarios.
5. Demostrar la seguridad, escalabilidad y compatibilidad de los SS7 abiertos.

1.4 Alcance

La implementación de OpenSS7 tiene como alcance, describir los pasos necesarios para instalar, compilar y adecuar, en un servidor Linux, los paquetes necesarios para que opere e interactúe entre si las pilas del protocolo SS7, en un ambiente aislado o también conocido bajo el término *stand-alone*.

Además, se presenta una explicación detallada de los diferentes componentes de cada nivel del modelo SS7 y SIGTRAN, encaminada a poder conectar la experiencia de la instalación de los paquetes, antes mencionados con la teoría de señalización.

Por otro lado, se comparte un análisis comparativo de las actuales soluciones de señalización propietarias versus el implementado, OpenSS7.

1.5 Limitaciones

Como principales limitaciones para el presente trabajo se tiene que las configuraciones requeridas en cada nivel del protocolo SS7 para que opere según aplicación instalada (ISUP, CAP, INAP, SCCP...), esto es, la definición de los parámetros requeridos en los archivos planos de los módulos de *Protocolo de Transmisión de Control de Flujo (Stream Control Transmission Protocol*, *SCTP*), *Capa de Adaptación de Usuario de Nivel 3 en la Parte de Transferencia de Mensajes (Message Tranfer Part Level 3 User Adaptation, M3UA*), SIGTRAN y STREAMS son mencionados y ubicados tanto en los archivos de configuración como en los directorios dentro del servidor, sin embargo la programación de los mismos no son cubiertos por este trabajo.

Así mismo, uno de los requisitos que exige esta implementación en cuanto a [5]:

- La arquitectura:
    - ix86, x86_64, ppc (*MCP 860*), ppc64.

- Sistema Operativo
    - CentOS, Debian, Fedora, Mandrake, Gentoo, Red Hat, SuSE, SLES, Ubuntu.
- Kernel del sistema operativo:
    - 2.4 a 2.6.
- Las *GNU toolchain* (herramientas usadas para crear paquetes de código fuente portables) requeridos para la recopilación y de desarrollo:
    - autoconf 2.63, automake 1.10.1, libtool 2.2.4, gettext 0.17, flex 2.5.33, bison 2.3.

1.6 Metodología

El desarrollo del presente trabajo se plantea bajo el enfoque de una investigación científica cualitativa del tipo descriptivo, puesto que: “La investigación cualitativa evita la cuantificación. Los investigadores cualitativos hacen registros narrativos de los fenómenos que son estudiados mediante técnicas como la observación participante y las entrevistas no estructuradas” [6]

Por lo ya expuesto, se pretende que este trabajo sirva de guía en la investigación de los futuros desarrollos en el amplio mundo de OpenSS7, debido a la narración de la implementación del servidor Linux con aplicaciones SS7.

De esta manera, y para cumplir los objetivos, en el siguiente capítulo se desarrolla el marco teórico: donde la literatura mencionada ahonda en los conceptos alrededor de los cuales gira todo este tema de tesis.

Una vez logrado enfocar el contexto, se inicia con la secuencia ordenada de los pasos necesarios para la implementación del servidor Linux con los paquetes de señalización, señalando todos los recaudos a tomar en cuenta durante el proceso.

En el siguiente capítulo, el tercero, se realiza una parada para aterrizar el proyecto OpenSS7 en nuestros días, esto servirá para alinearse a las necesidades actuales referentes a los sistemas abiertos.

Mientras que, en el capítulo cuatro, se realiza una valoración de los resultados, esto es, una revisión de lo obtenido y poder verificar las perspectivas con las que se desarrolló este trabajo.

Una vez alcanzado todo el análisis antes mencionado se elabora el informe final de tesis con las conclusiones logradas y las futuras líneas de investigación que puedan surgir del trabajo.

CAPÍTULO 2

2 FUNDAMENTOS TEÓRICOS

2.1 Unidades y niveles del modelo SS7

Para iniciar con la revisión de las funciones del protocolo SS7 [7], se debe anticipar que se lo realiza en una red convencional basada en *Multiplexación por División de Tiempo (Time Division Multiplexing, TDM)*.

Lo primero que debemos conocer es que una red SS7 es una red de máquina-a-máquina antes que una red usuario-a-usuario, que solo requiere una pequeña intervención humana debido a que la mayoría de los procesos son automatizados y no requieren control de un operador.

El SS7 es una tecnología basada en conmutación de paquetes, donde cada paquete contiene la información necesaria para encaminar datos a través de

la Red sin establecer una conexión directa con el destino. Una unidad de señalización no es más que un paquete, sin embargo, SS7 tiene aplicaciones que requieren de diferentes estructuras de paquetes y capacidades, por lo que se tienen las siguientes unidades de señalización:

- *Unidad de Señal de Llenado* (*USL*):

  Es la unidad de señal de nivel más bajo, que actúa como una bandera en redes SS7 basadas en TDM, y estas unidades son enviadas por la red cuando no están siendo enviados datos y la Red se encuentra en estado libre. Una USL también puede ser enviada como un reconocimiento a una unidad de señal previamente recibida (Figura 2.1).

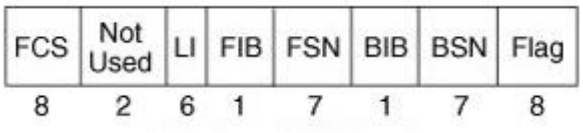


Figura 2.1. Campos de una unidad USL

- *Unidad de Señal de Estado de Enlace* (*USEE*):

  Se envía entre 2 puntos de señalización para indicar el estado del enlace de señalización, por lo tanto, la USEE es importante entre 2 puntos adyacentes. Así, cuando un enlace ha fallado, el punto de señalización que detecta el error es responsable de alertar al punto de señalización contiguo de la novedad. Los errores de este tipo son los que generan problemas de alineación.

La USEE consiste en los mismos componentes que el USL, con la adición del *Campo de Estado (Status Field, SF)*, campo que lleva la información de estado del enlace (Figura 2.2).

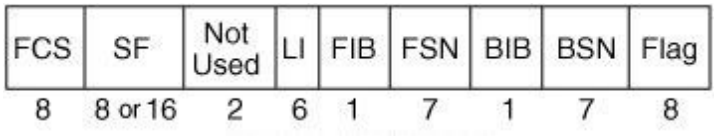


Figura 2.2. Campos de una unidad USEE

- *Unidad de Señal de Mensaje* (*USM):*

  Proporciona la estructura necesaria para transmitir todos los demás tipos de protocolos, como lo son ISUP, MAP, *Parte de Aplicación de Capacidades de Transacción (Transaction Capabilities Application Part, TCAP)*. La diferencia de esta unidad de las 2 anteriores, es que presenta 2 campos adicionales que son el *Octeto Indicador de Servicio (Service Indicator Octect, SIO)* y el *Campo de Información de Señalización (Signaling Information Field, SIF)*, el primero se usa en los mensajes de discriminación del nivel 3 para determinar el tipo de protocolo que está siendo presentado en la *USM*; mientras que el campo SIF contiene la etiqueta de encaminamiento y control de la información de los protocolos de nivel superior (Figura 2.3).

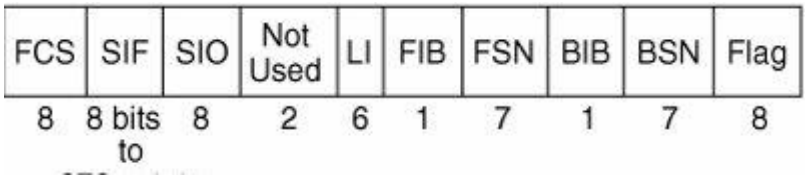


Figura 2.3. Campos en bits de una USM

2.1.1 Niveles de la arquitectura de SS7

Por otro lado, para lograr orden y escalabilidad, el estudio de SS7 se lo ha organizado a través de niveles, el cual difiere un poco del tradicional modelo de capas de *Interconexión de Sistemas Abiertos* (*ISO*), la primera diferencia es el uso del término nivel en lugar de capa, aun cuando se lo utilice en el mismo contexto.

Las funciones llevadas a cabo por los 4 niveles de SS7 es un símil de las llevada a cabo por las 7 capas del modelos ISO, pese a que algunas funciones de ISO carecen de propósito en una red SS7, sin embargo a lo largo del tiempo esta apreciación ha sido redefinida. En la Figura 2.4 se muestra una ilustración de este comparativo.

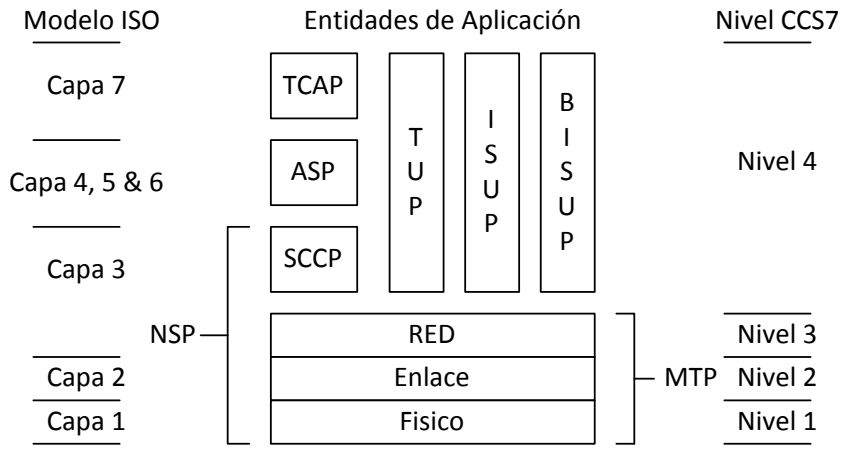


Figura 2.4. Los 4 niveles del protocolo SS7 vs el modelos ISO

Recordando que el enfoque es aplicable para redes TDM, es decir que para redes que verdaderamente son conmutadas por paquetes, como el *Protocolo de Control de Transmisión (Transmission Control Protocol, TCP)/Protocolo Internet (Internet Protocol, IP)*, no se usan las mismas técnicas. De esta manera se presentan las descripciones de los distintos niveles.

Nivel Físico

Similarmente al ISO, donde no se especifica una interfaz determinada, para el estándar *Telecordia* en Norte América y el estándar del *American National Standards Institute* (*ANSI*) en el resto del Mundo, es ampliamente aceptado el usar las interfaces DS0A o V.35.

Nivel de Enlace

Este nivel del protocolo SS7 provee la ya conocida función de detección/corrección de errores y entrega en secuencia de todos los paquetes de los mensajes SS7 de un nodo al siguiente adyacente. No provee encaminamiento sino solo los mecanismos necesitados para asegurar transferencia confiable de los datos sobre la red. En esta capa es usada la numeración secuencial para determinar si un paquete se ha perdido durante la transmisión; si esto ocurre se incrementa el contador de errores que es mantenido por el nivel 3. Después de una cantidad significativa de errores el enlace se activa como "fuera de servicio".

Nivel de Red

Las funciones que desempeña este nivel, que depende del nivel 2, son: encaminamiento, discriminación de mensajes y distribución.

La discriminación determina para quién va dirigido el mensaje, así, sí el mensaje va dirigido al nodo local, entonces se pasa a la distribución de mensajes, sí está dirigido a otro nodo, entonces el mensaje se pasa a la función de encaminamiento de mensajes, leyendo la información de los mensajes *calling-party* y *called-party*, para determinar a qué dirección física encaminar el mensaje.

La dirección física en redes SS7 se refiere al *Punto de Código* (*PC*) que se le asigna a cada nodo, el cual es único en cada red.

En este nivel hay 3 funciones de manejo de red: *Manejo de Enlace*, *Manejo de Ruta* y *Tráfico*.

El *Manejo de Enlace*, usa la USEE para notificar a los nodos adyacentes de problemas de enlace. Con esto el enlace queda fuera de servicio, con lo que no son transmitidos las USM's.

Nivel de Usuario, Parte de Usuario

Este nivel consiste en muchos protocolos, los mismos que son llamados *Parte de usuario y Parte de Aplicaciones*. Para las conexiones básicas de llamadas telefónicas se usa el protocolo *Parte de Usuario de Telefonía (Telephone User Part, TUP)* en Europa, o ISUP en América del Norte.

Para acceder a base de datos, el protocolo usado es TCAP especialmente para *Red Inteligente* y *Roaming* para redes inalámbricas.

Otro protocolo del nivel 4 es MAP*,* relativamente nuevo y usado en redes inalámbricas como el *Sistema Global para Comunicación Móvil (Global System for Mobile Communication, GSM)*. Cuya función principal es proveer un mecanismo que habilite la información del subscriptor inalámbricamente, para ser pasado de una red a otra.

Algo que se debe destacar, es que SS7 consta de dos partes funcionales, una es el manejo del contenido de los mensajes de señalización, es decir los

protocolos que anteriormente he mencionado, y la otra parte es la transferencia de los mensajes de señalización dentro de la red, o *Parte de Transferencia de Mensaje (Message Tranfer Part, MTP)*.

MTP-1

El *Nivel 1 de la Parte de Transferencia de Mensaje*, está ubicado en el nivel 1 de la jerarquía de SS7, la descripción física: *Signaling Data Link*, la cual consiste en un par de canales de transmisión digital de 64 Kbps [8]

MTP-2

Es el nivel 2 de SS7, y es el que provee de la detección/corrección de errores, secuencia de paquetes y de los indicadores de estados de enlaces y para su cometido usa las USL, USEE y USM.

MTP-3

Se refiere al nivel de red de SS7, el cual encamina los mensajes y depende de la entrega de mensajes de MTP2. Por otro lado, es la interfaz usada con los protocolos o aplicaciones de nivel 4, a través de *primitives*, que no son otra cosa que un método usado, a nivel de software, para pasar información al siguiente nivel del modelo jerárquico en cualquier dirección, ascendente o descendentemente [9]

2.1.2 Protocolos SS7

SCCP

Parte de Control de Conexión de Señalización (Signaling Connection Control Part, SCCP) es responsable de encaminamientos punto a punto ubicado en el nivel 4, que provee servicios orientados a conexión y no orientados a conexión, necesarios para soportar aplicaciones del siguiente nivel, además se ocupa de la gestión de base de datos.

El servicio que ofrece SCCP a capas superiores (aplicaciones y subsistemas) lo hace a través de los llamados *Números de Subsistemas (Subsystem Numbers, SSN)* prácticamente es la capa de transporte de los servicios basados en TCAP, como por ejemplo: Calling Card, Portabilidad, servicios de números gratuitos (1800, 800), *wireless roaming*, entre otros.

También es usado como un medio mediante el cual un STP puede desarrollar el encaminamiento a través de *Global Title Traslation* (*GTT*), explícitamente significa que el SSN y el punto de código de destino son alcanzados mediante un GTT (el mismo que no es más que un número en formato E.164).

Así como a nivel de MTP3 se encamina por *Punto de Código de Origen* (*Origination Point Code, OPC)* y *Punto de Código de Destino* (*Destination Point Code, DPC)*, en SCCP el encaminamiento se lo puede realizar por GTT (cuya información es el Called/Calling Party Address), SSN, OPC/DPC o una

combinación del último criterio con uno de los 2 primeros, cuando el direccionamiento se lo realiza por PC [10]

ISUP

El Protocolo para la *Parte de Usuario ISDN (ISDN User Part, ISUP)* está relacionado a circuito, usado en las llamadas telefónicas y encargado de establecer y mantener el circuito por el que cursa la llamada. Al ser un protocolo que soporta tanto circuitos digitales como analógicos es el remplazo del ya descontinuado protocolo TUP.

Dentro de la mensajería existen parámetros como:

- *Initiate Activate Message* (*IAM*): indica el inicio de una transacción ISUP e informa de un CIC reservado previo a pasar la llamada.
- *Circuit Identification Code* (*CIC*): encargado de identificar el circuito por donde pasaría una llamada.
- *Address Complete Message* (*ACM*): es el ACK del IAM, e informa que ha sido reservado el CIC que se solicitó en el IAM, y es enviada la alerta o timbrado al terminal receptor de la llamada.
- *Connect Message* (*CON*): con este mensaje se indica a la central llamante que se ha contestado la llamada.
- *Release Call* (*REL*): con este mensaje el lado que cuelga la llamada indica a la otra parte que ha culminado y deben volver a quedar libres los recursos como el CIC.
- *Release Complete* (*RLC*): el mensaje ACK para el REL.

SIGTRAN

SIGTRAN hace referencia a la pila de protocolos para transportar SS7 sobre Internet, que surge por la necesidad de acople de la señalización SS7, que antiguamente era transportada por una red TDM, ante la migración de los operadores telefónicos de sus redes a una infraestructura IP.

Así con el desarrollo y evolución de la redes IP, las redes SIGTRAN se han vuelto muy importantes en redes como *Voice Over IP* (*VoIP*), en las que se realiza la interfaz entre las denominadas *Redes Telefónicas Públicas Conmutadas (Public Switched Telephone Network, PSTN)* vía *Signaling Gateway* y los *Media Gateway Controller* (*MGC*) de las redes de VoIP (Figura 2.5).

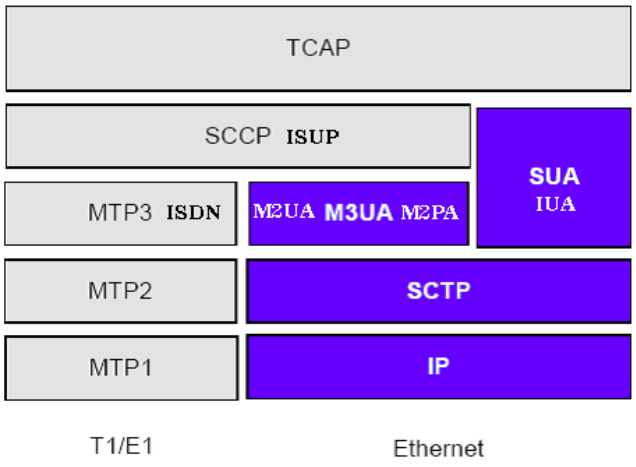


Figura 2.5. Modelo SIGTRAN vs modelo de SS7 sobre TDM

Como se puede notar en la figura algunos niveles son remplazados o adaptados en este nuevo modelo [7]

Interfaz para la Capa de Adaptador de Transporte de Tekelec Tekelec's Transport Adapter Layer Interface

De la *Interfaz para la Capa de Adaptador de Transporte de Tekelec (Tekelec's Transport Adapter Layer Interface, TALI)* se puede mencionar que la Red SS7 tradicional consiste en 3 tipos de dispositivos conectados a través de enlaces SS7 dedicados, estos 3 dispositivos primarios son:

- *Punto de Servicio de Señalización (Signaling Service Point, SSP*): que actúa como punto final de una red SS7 y es el originador de mensajes de señalización.
- *Punto de Transferencia de Señalización (Signaling Transfer Point, STP*): actúa como un switch, conmutando tráfico SS7 de un nodo a otro, es un concentrador de señalización.
- *Punto de Control de Señalización (Signaling Control Point, SCP)*: actúa como base de datos, guarda la información que usuarios necesitan acceder.

Además hay 3 tipos de enlaces SS7 dedicados:

- Enlaces de 56 Kbps (DS0, V35, OCU): estos enlaces los implementa los protocolos MTP-1 y MTP-2.

- DS1 High Speed Links: estos enlaces usan la *Capa de Adaptación de señalización para ATM (Signaling ATM Adaptation Layer, SAAL)* para proporcionar una alternativa a los enlaces de 56 Kbps.
- Enlaces E1: son los implementados por un canal de una E1 con una velocidad de 64 kbps.

En las redes SS7 convergentes, los dispositivos residen tanto en la Red tradicional PSTN como en Internet. Los servicios de SSP's, STP's y SCP's, pueden ser proporcionados por nuevos tipos de dispositivos de la red IP, sin embargo la red IP no está intencionada para remplazar a la Red tradicional PSTN, por ello la necesidad de que los dispositivos en los 2 tipos de redes deban ser capaces de comunicarse entre sí y convertir de una capa de protocolo inferior a la otra.

Los *Signaling Gateways* (*SG*) son los nuevos dispositivos que pueden también actuar como STP en una red convergente.

Los nuevos dispositivos de señalización en la red IP incluyen:

- *Media Gateway Controllers*
- *Media Gateways*
- *IP based SCPs*

TALI se puede utilizar entre 2 nodos para llevar el tráfico SS7 a través de TCP/IP.

Es la interfaz propuesta que ofrece SCCP, ISUP, y la encapsulación de mensajes MTP dentro de un paquete TCP/IP entre dos elementos de

conmutación. Además, TALI ofrece *SCCP Management* (*SCMG*), Primitivas MTP, el registro dinámico de los circuitos, y el envío de mensajes de control de llamadas basadas en la ubicación del circuito [12]

IUA

La arquitectura para transportar señalización *Red de Circuito Conmutado (Switching Circuit Network, SCN)* sobre IP, usa múltiples componentes, incluyendo un protocolo de transporte IP, un protocolo de transporte común de señalización y un módulo de adaptación para soportar los servicios esperados por un protocolo particular de señalización SCN, en el que el módulo de adaptación que es adecuado para el transporte de mensajes ISDN Q.921-User es IUA (*ISDN Q.921-User Adaptation Layer*) [13]

MTP2-User Peer-to-Peer Adaptation Layer

La Capa de Adaptación punto a punto para usuario MTP-2 (MTP2-User Peer-to-Peer Adaptation Layer, M2PA) es una nueva edición del comité SIGTRAN y está diseñado para el uso de nodos con conexión IP sin alguna conexión basada en TDM.

Un SG puede conectarse a otros nodos dentro de su propia red utilizando nada más que conexiones IP. Cuando se da este caso, un se asigna un PC al SG, y se comporta de la misma manera que un STP en una red SS7 convencional sin usar las facilidades TDM.

Entre las funciones de este protocolo están:

- Funcionalidad de MTP-2 e incluso soporta ciertas funciones de MTP-3.
- Mapeo de SS7 y entidades IP.
- Manejo de asociaciones SCTP.
- Retención de MTP-3 en la red SS7 [14]

MTP-2 User Adaptation Layer

La Capa de Adaptación de Usuario para MTP-2 (MTP-2 User Adaptation Layer, M2UA) fue desarrollada para ser usado como soporte para niveles más bajos del tráfico SS7 sobre IP. A nivel de transporte cuenta con SCTP.

El ejemplo más claro de uso es entre un SG y un MGC. En el SG, M2UA es responsable de mantener el estado de todos los *Procesos de Servidor de Aplicación (Application Server Processes, ASP)*, que han sido configurados en ese SG como direcciones de destino, ya que puede activarlos o desactivarlos, además en el caso de que exista más de un ASP para una determinada aplicación, se encarga de declarar al activo y los *standby*. Por otro lado, soporta 3 tipos de modos para los ASP: activo/inactivo, Carga Compartida *y broadcast.*

M2UA también soporta implementaciones cliente/servidor, lo que significa que la pila de protocolo puede funcionar por sí mismo en un servidor configurado ya sea como cliente o servidor.

Un aspecto que no es cubierto o al menos no está definido en M2UA es el control de flujo y congestión, estas funciones han sido relegadas a implementaciones individuales.

Entre los servicios proporcionados por M2UA están:

- Soporte para la interfaz de frontera entre MTP2 y MTP3.
- Soporte para la comunicación entre los módulos de manejo de capa en un SG y un MGC.
- Soporte para el manejo de asociaciones activas en SG y MGC [15]

MTP3 User Adaptation

Capa de Adaptación de usuario para MTP-3 (MTP3 User Adaptation, M3UA): describe la capa de adaptación de MTP3 en el modelo SIGTRAN. Utiliza el servicio de SCTP como el protocolo base de transporte de señalización para tomar las ventajas como:

- Entrega secuenciada de mensajes de usuarios (aplicaciones) dentro de múltiples *stream*s.
- Conmutación opcional de mensajes de usuarios en datagramas SCTP.
- Tolerancia a fallas a nivel de red, por soportar *multihome*.
- Resistencia a ataques de *flooding*.

Por otro lado, maneja los conceptos de ASP o *IP Server Process* (*IPSP*) y para éstos, proporciona el equivalente a *primitives* en su nivel superior a los usuarios MTP3. En este sentido, la capa ISUP y/o SCCP en un ASP/IPSP es inconsciente de que los servicios MTP3 esperados se ofrecen de forma

remota desde una capa MTP3 en un *Signaling Gateway Process* (*SGP*) y no por una capa MTP3 local.

Entre las funciones inter-redes de M3UA están:

- Indicar a los usuarios MTP3, en un ASP, que un destino no es alcanzable.
- Indicar a los usuarios MTP3, en un ASP, que un destino ahora es alcanzable.
- Indicar a los usuarios MTP3, en un ASP, que el mensaje a un destino está experimentando congestión.
- Indicar a los usuarios MTP3, en un ASP, que los caminos de destino están restringidas [16]

Capa de adaptación de Usuario SCCP

La *Capa de Adaptación de Usuario SCCP (SCCP User Adaptation, SUA)*, es el nivel que, en el modelo SIGTRAN, brinda una capa de adaptación de SCCP en SCTP, es decir que gracias a esta capa se pueden ofrecer los mismos servicios que se alcanzan con los mensajes de los usuarios SCCP, como TCAP, en un ambiente TDM, tales como:

- Es capaz de informar acerca de cambios de manera asíncrona.
- Traducción de GTT.
- Soporte a SCTP a través de la gestión de asociaciones.

- Soporta comunicaciones tanto orientadas a conexión como no orientadas a conexión.

Teniendo como principal desventaja el no poder transportar ISUP, para lo cual se usa M3UA [7]

Capa de Adaptación de usuario TCAP

TCAP-User Adaptation Layer (*TUA*) es el transporte para cualquier usuario de señalización TCAP como por ejemplo INAP, MAP..., sobre IP usando SCTP, este protocolo es modular y simétrico para permitir trabajar en diversas arquitecturas. Los elementos de protocolo son agregados para lograr una perfecta operación entre dominios SS7 e IP.

Los servicios que proporciona TUA son:

- Soporte para el transporte de mensajes de usuarios TCAP.
- Funciones de manejo o gerenciamiento nativo de TCAP.
- Soporte para poder pasar los mensajes de administración de SCCP.

Mientras que como una función principal tiene:

- Traducción de dirección y mapeo en el SG.
- Traducción de dirección y mapeo en el ASP.
- Mapeo de *Streams* SCTP [17]

SCTP

Nace a raíz de la necesidad de contar con un protocolo de transporte que brinde mejores garantías que su par TCP/IP, ya que:

- Ciertas aplicaciones requieren de transferencia confiable sin mantener la secuencia de paquetes o manteniéndola parcialmente, esta característica lo complica el *head-of-line blocking* de TCP.
- TCP tiene un alcance limitado de sockets, lo que dificulta el proveer alta disponibilidad en capacidad de transferencia de datos usando *multi-homed hosts*.
- SCTP desempeña el servicio de transferencia confiable dentro del contexto de asociación. En este sentido las conexiones SCTP son un concepto más amplio que las conexiones TCP.
- El paquete SCTP está compuesto de una serie de campos que permiten realizar el control de la conexión *end-to-end* mediante una asociación [18]

TCAP

La recomendación de la ITU-T Q.771-Q.775 describen a TCAP como un protocolo muy versátil usado para 2 propósitos esenciales: acceder a base de datos remotas e invocar características de una entidad de una red remota. Una de las características de TCAP es el uso de *Transaction ID's* para poder diferenciar las operaciones que se realiza sobre un mismo subsistema, una

analogía de esta capacidad es la que realiza TCP a través de *puertos* con una misma dirección IP.

A través de las bondades que ofrece TCAP, un abonado podría auto-aprovisionarse un determinado servicio desde su equipo terminal, en lugar de contactarse con el operador y solicitar este nuevo servicio.

Otro uso, ampliamente acogido por las operadoras móviles, es el servir de soporte o plataforma al protocolo MAP para que este transporte los mensajes cortos de textos, *Servicio de Mensajes Cortos (Short Message Service, SMS)*, servicio que durante muchos años fue un valor agregado que generaba significativos ingresos para las operadoras telefónicas, con el advenimiento de los servicios de datos y sus redes sociales, los SMS's fueron perdiendo protagonismo [7]

2.2 Sistemas de código abierto

Para que un sistema o software pueda ser considerado como un programa con licencia de código abierto no solo debe cumplir que su código esté accesible, sino también debe cumplir los principios básicos detallados en la *Open Source Definition*, que es una definición propuesta por el grupo *Open Source Iniciative* (organización fundada en 1998 dedicada a promover el código abierto), y que reza:

1. Libre redistribución.

2. Código abierto: todos los programas deben incluir el acceso a su código fuente, esto con el fin de realizar modificaciones.
3. Trabajos derivados: la licencia debe permitir modificaciones y trabajos derivados, y estos podrán ser redistribuidos bajo los mismos términos de la licencia del programa original.
4. Integridad del código fuente del autor: se puede restringir la distribución del código fuente modificado si es que la licencia permite la distribución de “archivos parches” junto al código fuente, con el propósito de modificaciones en el momento de la compilación.
5. No discriminación contra grupos o personas: con esta licencia no debe existir discriminación en la concesión.
6. No discriminación en contra de los fines perseguidos: la licencia no debe restringir a persona alguna de hacer uso del programa en un específico campo en el que se pretenda usar.
7. Distribución de la Licencia: los derechos que la licencia de un programa otorga, se los hereda a todos quienes les ha sido distribuida dicha licencia.
8. La licencia no debe ser específica a un producto: los derechos que la licencia de un programa distribuido no depende de ser parte de una específica distribución de software.

9. La licencia no debe restringir otro software: la licencia no restringe otros programas que se distribuyan juntos al programa con licencia de código abierto.
10. La licencia debe ser tecnológicamente neutral: ninguna disposición de licencia puede basarse en tecnología o uso de alguna interfaz específica [19]

2.2.1 Sistemas Operativo Linux

Linux es una combinación de una versión basada en unix, debido a su kernel o núcleo libre y el sistema GNU. Este sistema operativo, multiusuario y multitarea, es el desarrollo más importante del software libre o también llamado sistema operativo de *Código Abierto*.

En cuanto a la estructura, Linux, puede aislar al usuario del kernel a través del *Shell* (concha en español que indica el rol de protección) que es el intérprete de las órdenes e invocador de aplicaciones.

El *Shell,* como se indicó, es un programa intérprete de órdenes y que arranca con el sistema operativo, uno de los más difundidos es el *shell bash*, el mismo que fue creado para ser usado en el proyecto GNU.

Por otro lado, el kernel interactúa directamente con el hardware administrando y controlando todos los recursos del sistema, la versión del núcleo es actualizada para cada distribución de Linux, y en cualquier *shell* puede determinarse con la orden *uname –r*.

Todas estas particularidades llevan a que sea el sistema operativo más aceptado en servidores, dispositivos de bolsillo y teléfonos móviles [20]

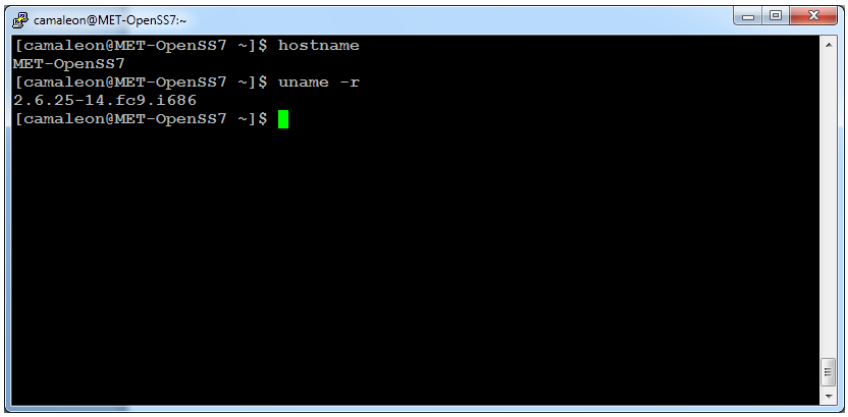


Figura 2.6. Servidor OpenSS7: Fedora 9 con kernel 2.6.25

2.2.2 Ideas generales de VoIP en Linux

El primer paso que se debe realizar para empezar a detallar la VoIP, es diferenciarla de la Telefonía IP, la cual es un concepto que puede verse limitado a aplicaciones o simplemente software en Internet que permitan realizar llamadas, mientras que VoIP no es algo tan simple ya que es una técnica que ofrece una gama de servicios de telefonía a través de protocolos (tanto como para señalización como para aplicación) y equipos integrados en Internet o de telefonía tradicional (Figura 2.7).

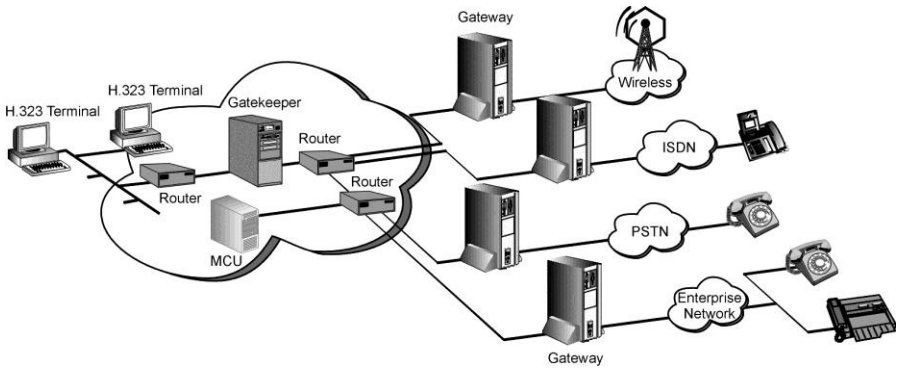


Figura 2.7. Solución Voz sobre IP

De esta manera, esta técnica se compone de arquitecturas de protocolos de señalización como H.323/*Protocolo de Iniciación de Sesión (Session Initiation Protocol, SIP)*, los cuales definen *Gateways* y *Gatekeepers*, plataformas encargadas de dar la interfaz con la red y establecer las conexiones y autenticaciones para conectar, encaminar y terminar llamadas.

Algunos de los protocolos de señalización (Figura 2.8) usados en VoIP son: H.323, SIP, *Media Gateway Control* (*MEGACO*) o también conocido como H.248, permiten:

- La negociación del tipo de codificador-decodificador (*CODEC*) a usar en la conexión de llamada.
- El intercambio y especificación de los números de puertos necesarios en el establecimiento de la llamada.
- La determinación del protocolo usado para transportar la voz [22]

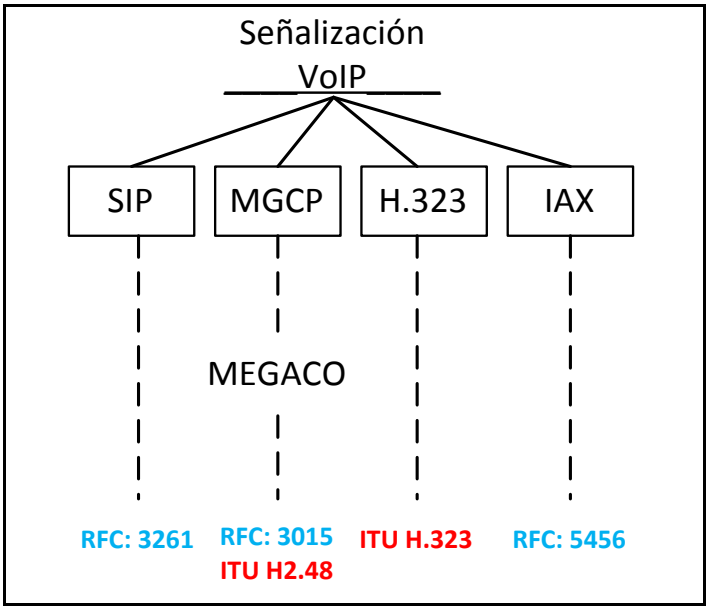


Figura 2.8. Protocolos de señalización en VoIP

Se considera que la versatilidad de los sistemas de VoIP es algo muy positivo dentro de los aspectos a tomar en cuenta al momento de decidirse por esta migración de arquitectura, ya que también pueden funcionar en sistemas operativos *de Código Abierto* como Linux, demandando solo una pequeña parte del costo que requeriría el implementar una solución con sistema operativo propietario [23]

Entre el software libre para el propósito de tener una *Private Branch eXchange* (*PBX*) que maneje VoIP se tiene a Ferrari, Dodge Omni y el, probablemente, más poderoso: Asterisk, el cual soporta muchos de los protocolos de VoIP, H.323, SIP, *Inter-Asterisk eXchange* (*IAX*), entre otros. Mediante Asterisk se puede construir algo tan simple como tener una máquina contestadora que graba y envía estos mensajes a una dirección de

correo electrónico hasta soluciones más sofisticadas como un sistema de comunicaciones con miles de abonados con un bajo costo [24]

CAPÍTULO 3

3 DESARROLLO DEL PROYECTO

3.1 Proyecto OPENSS7

Como ya se ha mencionado en la introducción, el proyecto OpenSS7 tiene como principal objetivo el brindar de manera robusta la pila del protocolo SS7, bajo licencia LPG para Linux y otros sistemas operativos UN*X, teniendo como misión tratar de responder a los impedimentos al uso extendido de SS7, tanto dentro como fuera de la comunidad portadora: gastos, complejidad, colaboración, certificación de competencias del *core* y experiencia [25]

Adicionalmente, el proyecto está involucrado en diferentes áreas específicas:

Aplicaciones

Las aplicaciones se refiere a los proyectos que usan la pila del protocolo SS7 antes que los componentes de la pilas SS7, estos proyectos incluye: GSM/MAP, *Registro de Ubicación Base (Home Location Register, HLR)*, *Servicio General de Paquetes de Radio (General Packet Radio Service, GPRS)*, *Centro de Servicios de Mensajes Cortos (Short Message Service Center, SMSC)*, *Red Inteligente (Intelligent Network, IN)*, *Portabilidad de Número Local (Local Number Portability, LNP)*, ENUM/*Puntero de Autoridad de Nombramiento (Name Authority Pointer, NAPTR)*, CS/AIN Call Model, OpenSwitch, integración de Asterisk PBX e Integración Kannel, este último es un compacto y poderoso Gateway de WAP y SMS Open Source.

Pila SS7

El principal campo de concentración del proyecto OpenSS7 son los componentes de la pila SS7 que proveen los diferentes niveles del protocolo SS7, que son: MTP Level 2, MTP Level 3, SCCP, TCAP, ISUP y *Protocolo de Control de Llamada de Portador Independiente (Bearer Independent Call Control, BICC)*.

Pila ISDN

Los componentes de la pila ISDN proveen los diferentes niveles del protocolo y estos incluyen: ISDN Q.931, Q.932, Q.933, *Procedimento Canal D de Acceso al Enlace (Link Access Procedure for D-channel, LAPD* o *Q.921)*,

Procedimento para Frame Relay de Acceso al Enlace (*Link Access Procedure for Frame Relay, LAPF* o *Q.922)*, *Procedimento de Acceso al Enlace Balanceado (Link Access Procedure, Balance, LAPB)*, X.25, así como los relacionados con IDLC, V5.2, GR-303-CORE, *Sistema de Señalización de Red Digital Privada (Digital Private Network Signalling System, DPNSS)*, *Sistema de Señalización de Acceso Digital, Digital Access Signaling System, DASS)*.

Pila SS7 IP

Los componentes de la pila SIGTRAN (SS7 sobre IP), proporcionan la base para la redundancia y distribución en el proyecto OpenSS7, comprende los programas: IPSS7, M2PA, M2UA, M3UA, SUA, TUA and TALI.

Pila VoIP

Los componentes de la pila VoIP proporcionan la señalización en el ambiente de VoIP, comprende: BICC, H.225.0 and SIP-T.

Media Gateway (MG)

Es uno de los más recientes enfoques del proyecto y los componentes de la pila, proveen la señalización en el ambiente MG y comprende: CH, MX, MG, MGCP and MEGACO.

Drivers de dispositivos SS7/ISDN

Los controladores de dispositivos de hardware realizan el control especializado para SS7, ISDN, SIGTRAN y VoIP, inclusive para la parte de canales de voz en Media Gateway puede ser desarrollado en ciertas tarjetas, las tarjetas usadas son: Sealevel ACB56 ISA (para V.35), la E400P-SS7 y T400P-SS7 quad E1/T1 span, y la E100P-SS7 yT100P-SS7 single E1/T1.

Transporte IP

El proyecto OpenSS7 se ha encargado de desarrollar el transporte IP para el STREAM de LiS (*Linux Stream*) que soporta todos los componentes de la interfaz IP, como lo es la interfaz de STREAM: *Interfaz Proveedor de Transporte (Transport Provider Interface, TPI)* para sockets TCP, *Protocolo de Datagrama del Usuario (User Datagram Protocol, UDP)* y RAW (entorno gráfico) de Linux nativo.

Sistemas Operativos

Las pilas del proyecto OpenSS7 son basadas en STREAMS, y el proyecto proporciona los paquetes tanto *Linux Stream* y *Fast-STREAM* usados por muchos fabricantes de pila de protocolo SS7 de Linux. También provee componentes fundamentales de redes para STREAMS y sus respectivas bibliotecas como: *libstreams*, *libxnet* y *libsocket* [26] (Figura 3.1).

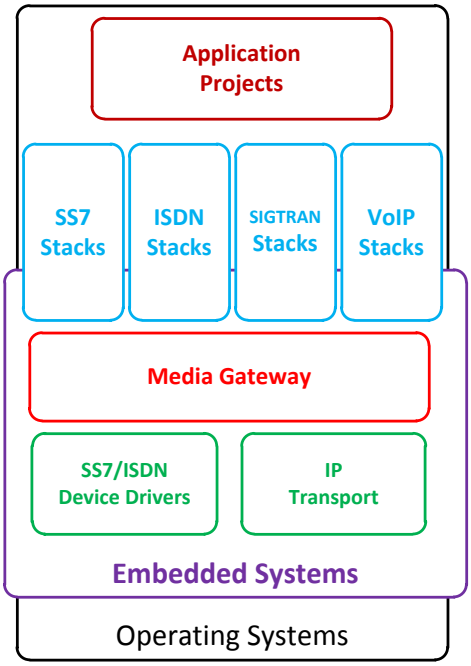


Figura 3.1. Áreas de cobertura del proyecto OpenSS7

STREAMS

Es un marco flexible para la comunicación entre un proceso a nivel de usuario y un driver del kernel que provee una interfaz kernel-usuario. STREAMS también proporciona un conjunto completo de funciones de utilidad del kernel para el desarrollo y la aplicación de módulos y controladores residentes en el núcleo. STREAMS provocó la especificación DDI/DKI, la cual es una arquitectura independiente del controlador de la interfaz del kernel que proporciona un conjunto estándar de funciones del kernel (más allá de sólo STREAMS) para el desarrollo de controladores de dispositivos y software [27]

El kernel de Linux no fue pensado para el uso de STREAMS, sin embargo la implementación de componentes como TCP/IP y SCTP en el kernel, han provocado que STREAMS, gracias a sus bondades y alcance, tenga un papel protagonista en el desarrollo de aplicaciones de telecomunicaciones y en implementación de protocolos del modelo ISO, permitiendo de esta manera fortalecer al sistema operativo Linux y convirtiéndolo en una plataforma capaz de soportar aplicaciones de grandes envergaduras, como las de telefonía [27]

3.2 Linux como Sistema Operativo para operar SIGTRAN

El primer bastión a colocar en la pila SIGTRAN es el protocolo SCTP, el cual, como se vio anteriormente, es un protocolo de transporte que ofrece ciertas ventajas frente a sus pares UDP y TCP, estas ventajas, como robustez, disminución del retraso (*delay*), mayor seguridad contra *SYN-flood*, mayor flexibilidad, entre otras, han permitido satisfacer las altas exigencias de SIGTRAN [28]

Adicionalmente, tomando en cuenta que Linux es un sistema operativo de *Código Abierto*, lo que lo ha convertido en ocasiones en un excelente laboratorio para nuevos protocolos, permite a Randall Steward, coinventor de SCTP, iniciar el proyecto *Linux Kernel SCTP*, o también llamado LKSCTP, cuyo objetivo primordial era el de proporcionar una implementación tal de kernel de Linux, que haga referencia a SCTP, es decir lograr integrar SCTP

en el kernel de Linux, así mismo bajo licencia LPG GNU, obteniendo el primer *release* en Enero del 2001.

Actualmente LKSCTP es parte de la serie 2.6x del kernel de Linux y está siendo activamente mantenido y ampliado [29]

Por lo antes mencionado, en el desarrollo del presente trabajo, se ha optado por usar la distribución de Linux Fedora 9, es decir un Linux con versión de kernel 2.6.25. Con lo que se asegura la compatibilidad al instalar y operar paquetes del tipo *Administrador de Paquetes Red Hat (Red Hat Package Manager, RPM)*, que es una herramienta de administración de paquetes para GNU/Linux, de OpenSS7 tales como *strsctp*: STREAM SCTP.

3.3 Seguridad de SS7 en Linux

La seguridad de SS7 en Linux queda restringida a la seguridad que puede ofrecer el sistema operativo anfitrión, en este caso Linux, y la que ofrecen como potencialidades los diferentes protocolos en la pila de SS7 ya sea en redes TDM como SIGTRAN en IP.

La seguridad que ofrecen los protocolos es un tema importante en el desarrollo de aplicaciones, pues dependiendo de la seguridad que garantice cada protocolo una aplicación puede tener mayor o menor alcance, por ejemplo si no fuera por las funciones de seguridad *Autenticación, Autorización y Auditoría* (*AAA*) de Diameter, éste no podría ser una opción como protocolo de cobro de servicios de datos en una *Red Inteligente* en la

telefonía móvil, y a su vez Diameter, en transporte, se apoya de la seguridad de protocolos como SCTP o TCP en lugar UDP, además de *Seguridad en Protocolo Internet (Internet Protocol Security, IPSEC)* o *Seguridad en Capa de Transporte (Transport Layer Security, (TLS)*.

Mientras que para lo concerniente a la seguridad del sistema operativo, ciertamente se conoce que Linux puede ofrecer una gran seguridad, sin embargo es recomendable reforzar estas seguridades inherentes para brindar no solo seguridad sino estabilidad y resistencia a los ataques, el camino es algo denominado *hardening* del sistema operativo, proceso que podría sufrir una ligera variación dependiendo de la aplicación que esté funcionando en el servidor, pudiendo ser un e-mail, un servidor *Protocolo de Transferencia de Hyper Texto (Hyper Text Transfer Protocol, HTTP)* o incluso un servidor de OpenSS7. Además, el *hardening* del sistema operativo Linux que se lo puede realizar de manera manual como también usando herramientas *de Código Abierto* para esta finalidad, en ambos casos lo que se busca es mitigar los riesgos y vulnerabilidades, a través de actualizaciones del sistema, deshabilitando servicios innecesarios, bloqueando puertos que no se usen, usuarios personalizados... Los programas de *Código Abierto* ayudan a los administradores a automatizar los procesos antes mencionados, uno de ellos es el *Bastille* así como también los programas antivirus [30]

3.4 Requisitos para montar el proyecto

En el capítulo 1 se realizó una revisión muy breve en cuanto a los requisitos del sistema para realizar la implementación de la pila SIGTRAN y/o SS7 en un servidor Linux, es decir, para llevar a cabo la implementación de un servidor OpenSS7; en este apartado se da una explicación más somera en cuanto a los requisitos básicos tanto en Hardware como en software.

3.5 Software

En lo referente al software [5], OpenSS7 considera al sistema operativo como un requisito para que la implementación tenga éxito de acuerdo a las pruebas por ellos realizadas. Por tal motivo a continuación se lista todas las versiones y distribuciones de Linux cuyo kernel puede ser versión 2.4 o 2.6. Esto debido a que el SCTP integrado en estas versiones les proporciona la compatibilidad requerida en la instalación y operación de los STREAMS 0.7a.4 del proyecto:

- CentOS Enterprise Linux 3.4 (*centos34*), 4.0 (*centos4*), 4.92 (centos49), 5.0 (*centos5*), 5.1 (*centos51*) y 5.2 (*centos52*).
- Debian: 3.0r2 Woody (*deb3.0*), 3.1r0a Sarge (*deb3.1*), 4.0r1 Etch (*deb4.0*), 4.0r2 Etch (*deb4.0*) y 4.0r3 Etch (*deb4.0*).
- Fedora: core 1 (*FC1*), Core 2 (*FC2*), Core 3 (*FC3*), Core 4 (*FC4*), Core 5 (*FC5*), Core 6 (*FC6*), Core 7 (*FC7*), Core 8 (*FC8*) y Core 9 (*FC9*).
- Gentoo: 2006.1 (no probado) o 2007.1 (no probado).

- Lineox: 4.026 (*LEL4*) o 4.053 (*LEL4*).
- Mandrakelinux: 9.2 (*MDK92*), 10.0 (*MDK100*) o 10.1 (*MDK101*).
- Mandriva: Linux LE2005 (*MDK102*), Linux LE2006 (*MDK103*) o One (no probado).
- RedHat: Linux 7.2 (*RH7*), Linux 7.3 (*RH7*), Linux 8.0 (*RH8*), Linux 9 (*RH9*), Enterprise Linux 3.0 (*EL3*), Enterprise Linux 4 (*EL4*) o Enterprise Linux 5 (*EL5*).
- SuSE: 8.0 Professional (*SuSE8.0*), 9.1 Personal (*SuSE9.1*), 9.2 Professional (*SuSE9.2*), OpenSuSE (*SuSEOSS*), 10.0 (*SuSE10.0*), 10.1 (*SuSE10.1*), 10.2 (*SuSE10.2*), 10.3 (*SuSE10.3*) o 11.0 (*SuSE11.0*).
- SLES: 9 (*SLES9*), 9 SP2 (*SLES9*), 9 SP3 (*SLES9*) o 10 (*SLES10*).
- Ubuntu: 5.10 (*ubu5.10*), 6.03 (*ubu6.03*), (*ubu6.10*), 7.04 (*ubu7.04*), 7.10 (*ubu7.10*) o 8.04 (*ubu8.04*).
- WhiteBox: Enterprise Linux 3.0 (*WBEL3*) o Enterprise Linux 4 (*WBEL4*).

3.6 Hardware

En cuanto al hardware, se empieza señalando que las arquitecturas que son soportadas son:

- ix86.
- x86_64.

- ppc (*MCP 860*).
- ppc64.

Cabe destacar que no es necesaria la arquitectura de *Intel*. Por otro lado, los paquetes OpenSS7 requieren de alrededor de 1MB de memoria del Kernel, por consiguiente no es necesario más allá de 32 MB de memoria [5]

En el caso de una implementación ISUP, por ejemplo, sobre su correspondiente capa, se necesitaría un hardware adicional que permita realizar la interfaz para la interconexión, en este caso se podría usar una tarjeta V401P, la cual provee 4 puertos RJ45 para la conexión de E1/T1's. También existe una gama adicional de tarjetas que el proyecto OpenSS7 ha homologado según la necesidad y exigencia, y estas son [31]:

- Interfaz de Tarjeta V401P: del fabricante Varion, esta tarjeta trae integrado 4 conexiones para E1's o T1's, una de sus grandes ventajas es un bajo costo y su alta compatibilidad con PC's pero como bajo desempeño.
- Interfaz de Tarjeta A101/102/104c: de acuerdo a tarjeta usada, se puede contar con 1-, 2-o 4- puertos para E1's, T1's o J1's, su fabricante es Sangoma.
- Interfaz de Tarjeta TE405/410: son tarjetas PCI que permiten así mismo la conexión de hasta 4 E1/T1´s, fabricadas por Digium, su ventaja es el costo y que corre a alta velocidad de buses, sin embargo no puede alcanzar un gran desempeño.

- Interfaz de Tarjeta V400P: similar a la V401P Interface Card.
- Interfaz de Tarjeta T/E100P: tarjeta PCI que proporciona tan solo un puerto para E1/T1, y tiene el más bajo desempeño a nivel de señalización.
- Tormenta Interface Card: es una tarjeta ISA de 2 puertos T1's, fabricada por Zapata Telephony.
- Interfaz de Tarjeta CPC388: esta tarjeta brinda 8 puertos E1/T1/J1 con puertos duales Ethernet en una tarjeta Compact PCI PICMG 2.16.
- Interfaz de Tarjeta CPC396: proporciona 1 puerto T3 con puertos duales Ethernet en una tarjeta Compact PCI PICMG 2.16., es fabricada, como la anterior, por *Perfomance Technologies.*
- Interfaz de Tarjeta PCA200E: es una tarjeta *Modo de Transferencia Asíncrono (Asyncronous Transfer Mode, ATM)* a 155 MHz, fabricada por Fore, tiene gran desempeño a nivel de señalización, su desventaja es que no soporta los tradicionales enlaces SS7 y requiere de conexiones o switches externos ATM.

3.7 Proceso de implementación

Para el proceso de implementación se debe especificar el hardware utilizado, que para nuestro caso, se usó una computadora personal de escritorio:

- HP Compaq DC 7700:
    - Arquitectura i686.

- Memoria RAM 1,5 GB.
- 250 GB de Disco duro.
- Procesador Intel Core2Duo E2160.

Figura 3.2. PC usada en proyecto OpenSS7

Mientras que el sistema operativo, como ya se lo mencionó anteriormente, se trabajó con una distribución de Linux Fedora Core 9 con Kernel 2.6.25 (Figura 3.3).

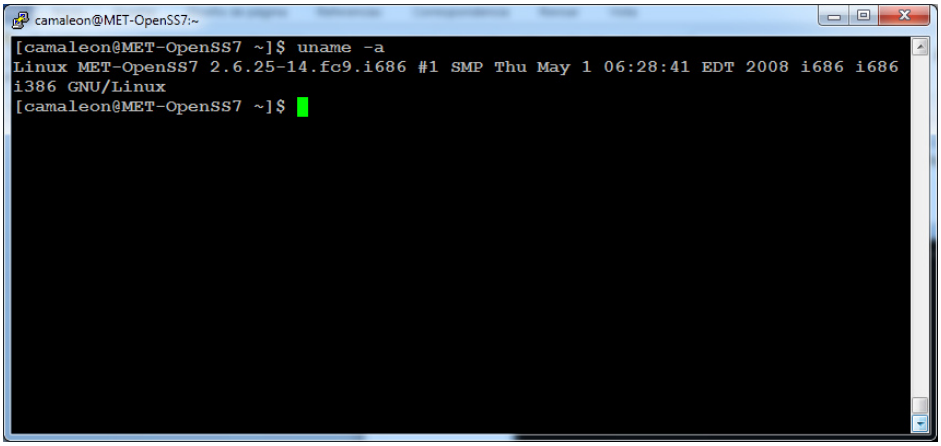


Figura 3.3. Especificación de Sistema operativo

La instalación de una versión Fedora actualizada se la puede realizar bajando los archivos desde el sitio oficial www.fedora.org, sin embargo en el caso de necesitar una versión anterior se la obtiene de los repositorios de los siguientes sitios recomendados:

- *How to Forge* [32]
- *Ros Business Consulting* [33]

Los usuarios en el sistema son: root (el súper usuario) y camaleón (usuario con el que se realizan las instalaciones) tal como se muestra en la Figura 3.4.

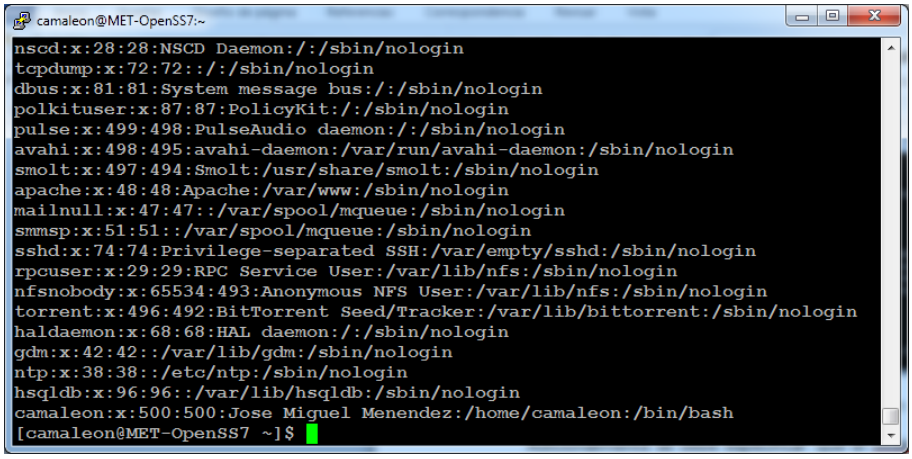


Figura 3.4. Usuarios creados en el sistema

Adicionalmente se debe especificar que el desktop es un GNOME (Figura 3.5).

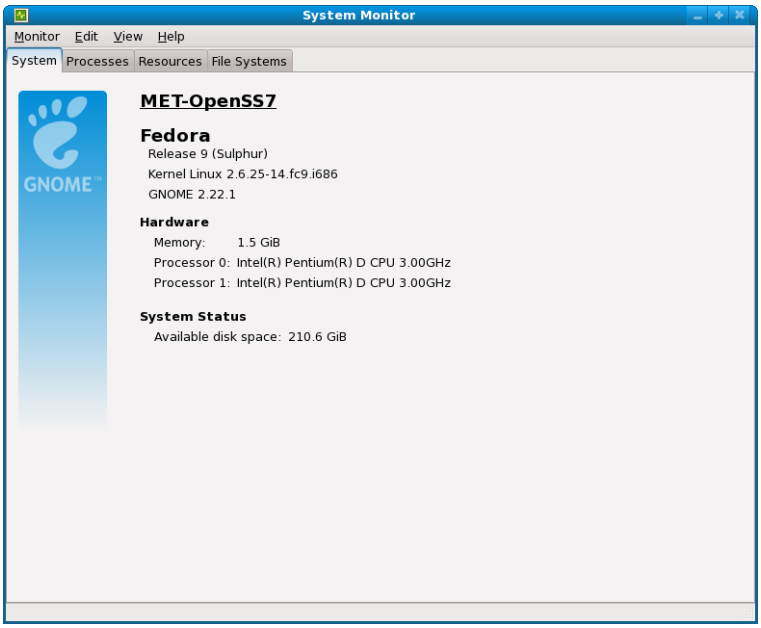


Figura 3.5. Apariencia del sistema operativo instalado

Sobre este sistema se realiza la descarga de los paquetes de la pila de OpenSS7, los cuales deben ser compilados e instalados.

Entorno de seguridad en la preinstalación

Todos los paquetes de software del proyecto OpenSS7 (www.openss7.org) son firmados con una clave privada creada bajo *Algoritmo de Firma Digital (Digital Signature Algorithm, DSA)* de 2048 bits, y su pareja, la clave pública se encuentra en el sitio oficial de Open SS7: *Public Keys* [34], desde donde debe ser importada.

Así mismo, la nomenclatura del nombre de cada paquete obedece a <*nombre_del_paquete*>*.tgz* y la firma de cada uno de ellos vienen dado en el archivo <*nombre_del_paquete*>*.tgz.asc*. Por lo tanto, se debe crear la pareja de claves, pública y privada para el sistema local, siendo de encriptación DSA de 2048 bits para poder manipular los paquetes provenientes de OpenSS7.

La creación del par: clave privada y clave pública, se lo realiza, en el sistema, a través de la orden mostrada en la Figura 3.6.

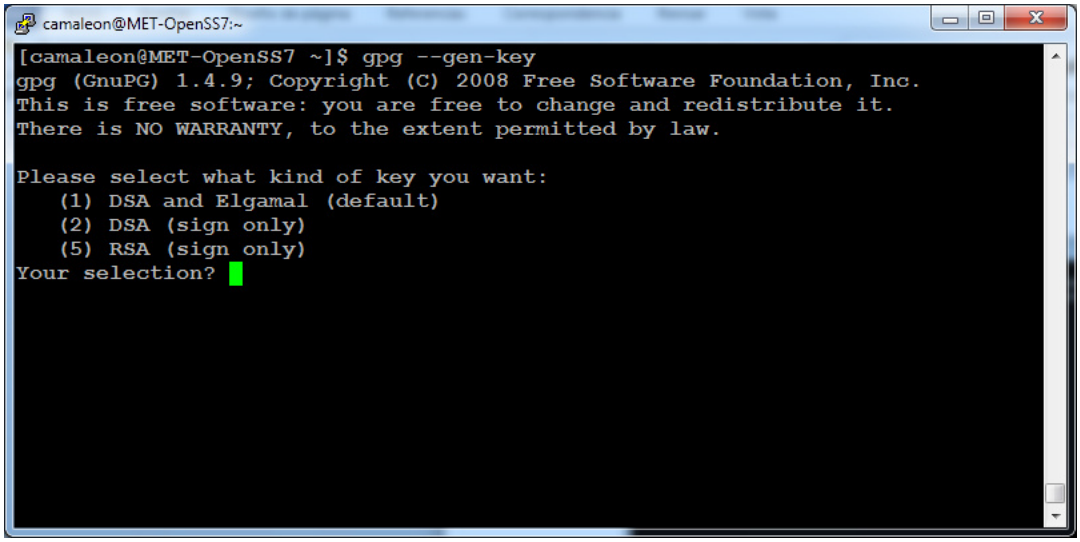


Figura 3.6. Creación de las claves pública y privada

Así, y de manera interactiva, se crea la pareja de claves de encriptación DSA de 2048 bits en el sistema local. Este proceso, también conocido como ceremonia de intercambio de claves, debe completarse importando la clave pública de OpenSS7 Corporation a nuestro sistema mediante la orden *gpg --import <archivo_clave_publica>.tgz.asc*, en la que el archivo con la información de la clave pública se llama *archivo_clave_publica*, el mismo que previamente fue descargado a nuestro servidor vía web o ftp. Una vez completado el proceso antes descrito, se está listo para, por motivos de seguridad, poder verificar la originalidad de los paquetes OpenSS7 (Figura 3.7).

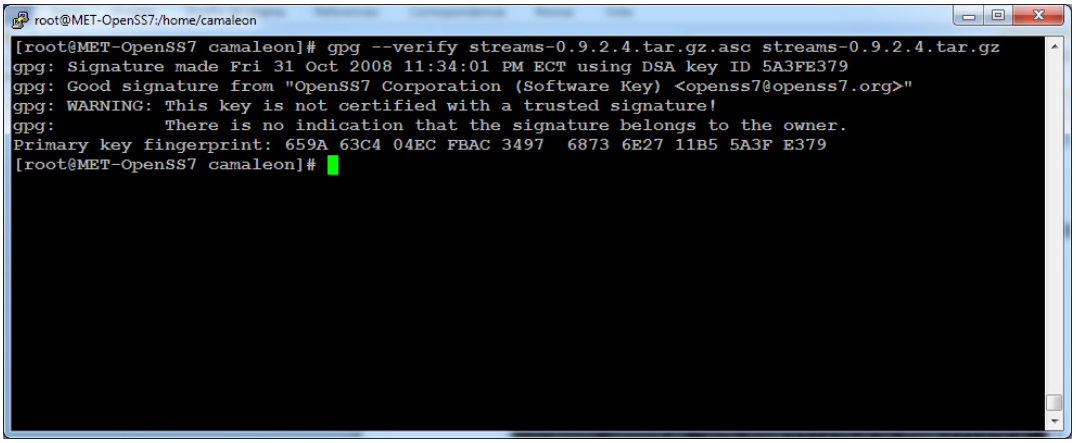


Figura 3.7. Verificación de legitimidad de los paquetes OpenSS7

A manera de ejemplo la Figura 3.7, muestra el paquete a instalar *streams-0.9.2.4.tar.gz* y su respectiva clave es la contenida en el archivo *streams-0.9.2.4.tar.gz.asc*, de esta manera nos aseguramos que estamos instalando un paquete emitido ciertamente por OpenSS7 Corporation.

Descarga, Compilación e instalación de Paquetes OpenSS7

Una vez realizada la verificación podemos proceder con la instalación de los paquetes, para los cuales, de manera general se debe ejecutar la siguiente secuencia de órdenes que permiten descargar, configurar, compilar, revisar, instalar, validar, desinstalar y eliminar cada uno de los paquetes.

Mediante la herramienta para descargas desde la web, wget, se procede con la descarga del paquete deseado:

```
[camaleon@MET-OpenSS7 ~]$ wget
http://www.openss7.org/tarballs/paquete_a_instalar.tar.bz
2
```

Dado a que el paquete de software viene comprimido, se procede a descomprimirlos con:

```
[camaleon@MET-OpenSS7 ~]$ tar -xjvf
paquete_a_instalar.tar.bz2
```

Se crea una carpeta o directorio a la que se denomina *build* y donde se guardará en la pila de “indexación” mediante *pushd*:

```
[camaleon@MET-OpenSS7 ~]$ mkdir build
[camaleon@MET-OpenSS7 ~]$ pushd build
```

Antes de la compilación del paquete, se debe realizar la autoconfiguración mediante la ejecución del script *configure*:

```
[camaleon@MET-OpenSS7 ~]$ ../paquete_a_instalar/configure
--enable-autotest
```

El resultado del paso anterior arroja la creación del archivo *make*, si algo resultó erróneo no tendríamos dicho archivo y no se podría continuar con la compilación:

```
[camaleon@MET-OpenSS7 ~]$ make
[camaleon@MET-OpenSS7 ~]$ make check
```

A continuación se realiza la instalación y la revisión:

```
[camaleon@MET-OpenSS7 ~]$ sudo make install
[camaleon@MET-OpenSS7 ~]$ sudo make installcheck
```

Los pasos siguientes pertenecen al proceso de *rollback* de lo realizado:

```
[camaleon@MET-OpenSS7 ~]$ sudo make uninstall
[camaleon@MET-OpenSS7 ~]$ popd
[camaleon@MET-OpenSS7 ~]$ sudo rm -rf build
[camaleon@MET-OpenSS7 ~]$ rm -rf paquete_a_instalar
[camaleon@MET-OpenSS7 ~]$ rm -f
paquete_a_instalar.tar.bz2
```

Sin embargo, sí las instrucciones detalladas arriba pudieran presentar problemas al momento de la compilación de los paquetes, proceso que, a manera de una revisión breve, se debe mencionar que es el responsable de tomar un programa fuente escrito en un lenguaje determinado (C para el caso de los paquetes OpenSS7) y crear un programa ejecutable binario en el lenguaje de la máquina en la que se está compilando el paquete. Para nuestro caso, el compilador usado es el denominado *Compilador Colector GNU (GNU Compiler Collection, GCC)* el cual fue desarrollado por el proyecto GNU y destinado para C, C++, Objetive C y Fortran [35]

El error más frecuente en la compilación de los paquetes a instalar, es que el compilador GCC no sea la misma versión con la que fue construido el kernel, por lo que, para poder determinar la versión del compilador con la que fue construido el kernel y la versión de compilador que trabaja en nuestro sistema se debe ejecutar lo mostrado en la Figura 3.8.

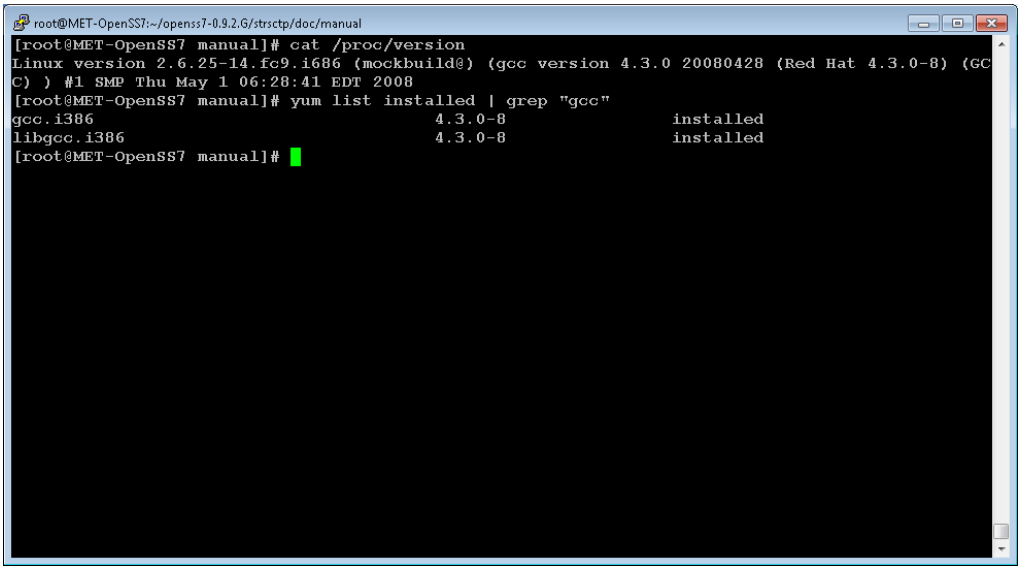


Figura 3.8. Revisión de versión de compilador GCC de Linux

Como se puede notar, con la primera orden se presenta información del kernel del sistema, entre ella la versión del compilador con el que fue construido y con la siguiente orden se anota la versión del compilador GCC instalado que opera en el sistema.

Instalación alternativa de paquetes de software en Fedora

Una manera diferente de realizar la instalación de paquetes de software en el sistema operativo Fedora es a través de la herramienta *yum*, la cual instala automáticamente, desde un repositorio declarado, en este caso OpenSS7 Corporation, todos los paquetes de la pila SS7 a instalar en Linux.

Para poder proceder de esta manera, como se mencionó, primero se debe declarar un archivo repositorio de OpenSS7 en el directorio

/etc/yum.repos.d/, el cual se lo descarga desde el repositorio oficial, ejecutando:

```
[camaleon@MET-OpenSS7 ~]$
REPOS="http://www.openss7.org/repos/rpms"
[camaleon@MET-OpenSS7 ~]$ wget
$REPOS/rpms/fedora/9/i686/repodata/openss7.repo
[camaleon@MET-OpenSS7 ~]$ sudo cp -f openss7.repo
/etc/yum.repos.d/
[camaleon@MET-OpenSS7 ~]$ sudo yum makecache
```

Una vez definido el repositorio, OpenSS7 pone a disposición una serie de paquetes virtuales que son muy fáciles de instalar y remover del módulo del kernel y bibliotecas, por ejemplo para instalar la última versión del paquete *dia* (el equivalente al software *visio* de microsoft) Figura 3.9.

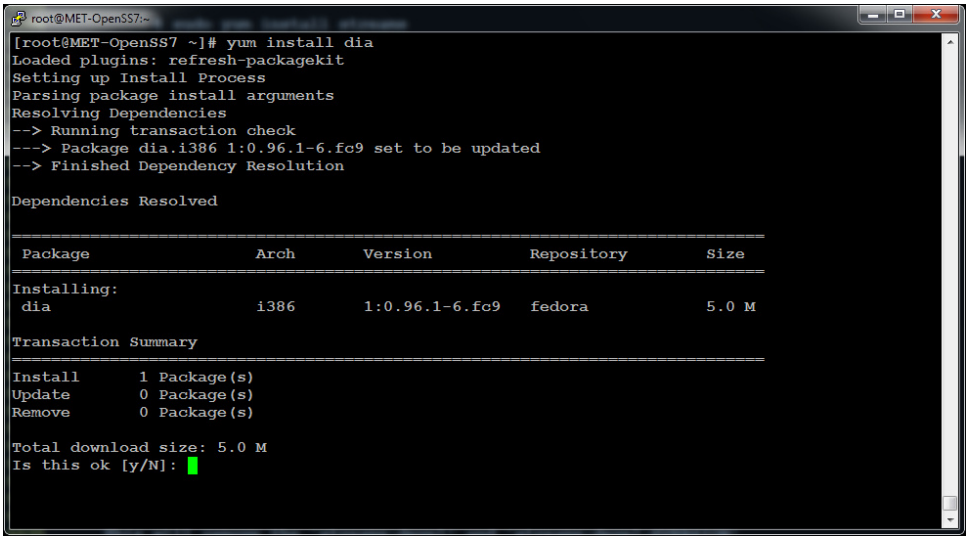


Figura 3.9. Instalación de un paquete a través de *yum*

Como se puede notar en la imagen, se intenta instalar el software *dia*, mediante la ejecución de: <*yum install dia*>, lo que provoca que el sistema

realice un búsqueda en los repositorios declarados (el servidor local debe alcanzar a los repositorios remotos que para el caso de OpenSS7 se requiere una conexión Internet), y en el caso de encontrar un paquete con ese nombre en uno de esos repositorios, indicaría al usuario de haberlo encontrado y además mencionaría las dependencias que se necesitan instalar previamente a la instalación del paquete principal. Hay que tomar en cuenta que para los paquetes de OpenSS7 se debe realizar la validación de firma para asegurarnos el estar instalando programas seguros.

Paquetes a instalar

Los paquetes de software que deben ser instalados en el servidor local son los necesarios para garantizar:

- La compatibilidad entre el sistema operativo y los propios paquetes, entre las diferentes capas de la pila SS7/SIGTRAN/VoIP.
- Las funcionalidades y aplicaciones de los componentes de las capas o niveles en el modelo de señalización SS7.
- La correcta operación de los controladores requeridos en el sistema.

OpenSS7 Corporation lista una serie de paquetes que pueden ser instalados de acuerdo a la necesidad o servicio requerido, sin embargo no cualquier paquete podría ser instalado sin respetar una cadena de dependencia, por ende y de acuerdo a lo experimentado, a continuación detallo la secuencia de paquetes instalados en el servidor local:

-*Stream*: aplicación de OpenSS7 de STREAMS SVR4.2 (*System V Release 4.2*), los streams son los que permite la comunicación entre un proceso a nivel de usuario y los controladores del kernel, es la parte primordial en SS7. Paquete a instalar: *streams-0.9.2.4*.

- *Strcompat*: proporciona una implementación de módulos de compatibilidad para AIX, HPUX, IRIX, LfS, MacOT, OSF/1, SVR 3.2, SVR 4.2, y UnixWarepara el uso con los paquetes LiS y Linux Fast-STREAMS. Paquete a instalar: *strcompat-0.9.2.7*.

- *Strxns*: aplicación de la X/Open XNS de servicios de red XNS, para CDI, DLPI, NPI y TPI. Paquete a instalar: *strxns-0.9.2.7*.

- *Strxnet*: aplicación de OpenSS7 de la biblioteca *libnet* de X/Open XNS/XTI. Paquete a instalar: *strxnet-0.9.2.12*.

- *Strsock*: implementación de bibliotecas *libsockets* de X/Open XNS Sockets, proporciona el módulo *sockmod*, el driver *socksys* y la biblioteca *libsocket*, solo aplica para la instalación de *Linux Fast-STREAMS* por el uso de sockets. Paquete a instalar: *strsock-0.9.2.4*.

- *Strinet*: este paquete implementa STREAMS sobre sockets, proporcionando acceso a las TPI a través de las líneas X/Open XNS/XTI para TCP, UDP y sockets IP. Paquete a instalar: *strinet-0.9.2.7*.

- *Strsctp*: es la implementación del protocolo SCTP. Paquete a instalar: *sctp-0.2.27*.

- *Strchan*: brinda los STREAMS Channels, lo que proporciona la multiplexación y controladores de canales para las tarjetas TDM usadas en OpenSS7. Paquete a instalar: *strchan-0.9.2.4*.

- *Strx25*: implementación de STREAMS para X.25 y Frame Relay. Paquete a instalar: *strx25-0.9.2.1*.

- *Strtty*: mediante este paquete se puede usar terminales (*tty*) basados en STREAMS. Paquete a instalar: *strtty-0.9.2.4*.

- *Iperf*: herramienta para pruebas que mide el desempeño de Internet, modificada para usar con el paquete Linux-sctp. Paquete a instalar: *iperf-2.0.8*.

- *Striso*: es la implementación de STREAMS OSI. Paquete a instalar: *striso-0.9.2.4*.

- *Netperf*: herramienta para pruebas que mide el desempeño de Internet, modificada para usar con los paquetes Linux-sctp, strinet, strxnet, y strsctpnet. Paquete a instalar: netperf-2.3.7.

- *Strisdn*: este es una implementación de STREAM OPENSS7, proporciona un controlador ISDN Q.931, siguiendo la interfaz de control de llamadas. Paquete a instalar: *strisdn-0.9.2.4*.

- *Strvoip*: implementación de STREAM OPENSS7 de varios componentes de la pila de VoIP, entre ellos H.245 y SIP. Paquete a instalar: *strvoip-0.9.2.4*.

- *Sigtran*: implementación de STREAM OpenSS7 de SIGTRAN incluyendo componentes como M2PA, M2UA, M3UA, SUA, TUA, IUA, DUA, V5UA, GR303UA y utilidades asociadas. Paquete a instalar: *sigtran-0.9.2.4*.

- *Strss7*: brinda los componentes de la pila SS7, ISDN, SIGTRAN y VoIP, al igual que la implementación proporcionada por los paquetes strss7, strisdn, sigtran strvoip por separado. Paquete a instalar: *strss7-0.9a.8*.

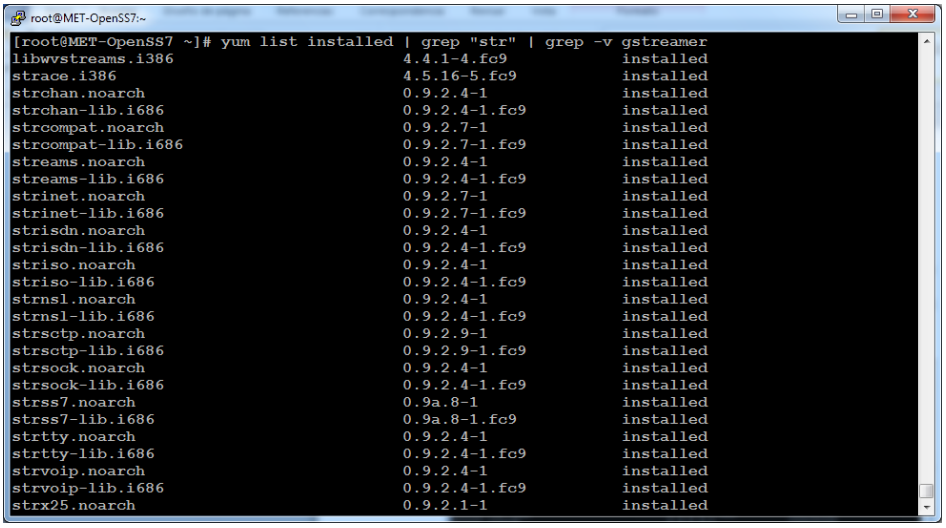


Figura 3.10. Paquetes OpenSS7 instalados

En la Figura 3.10 se muestra algunos de los paquetes de software que se encuentran en el servidor, en la consulta que se realiza con la orden *yum list installed* se filtra los que contengan la cadena *str* y que además no se trate de los paquetes *gstreamer*.

En otro aspecto, así luce el directorio */root/openss7-0.9.2.G* con los subdirectorios creados luego de instalar los paquetes mencionados (Figura 3.11).

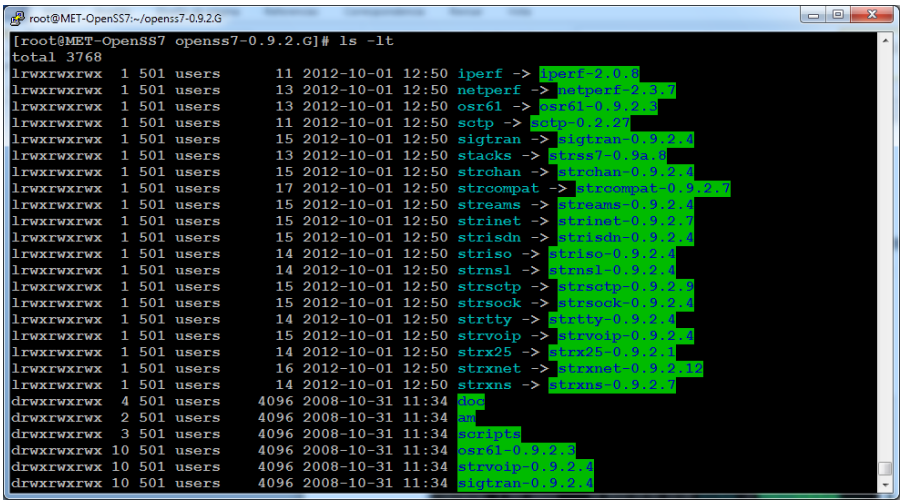


Figura 3.11. Directorios y subdirectorios creados por los paquetes de software OpenSS7 instalados

Dentro de cada subdirectorio, que representa a cada uno de los paquetes instalados, se puede encontrar información relativa a documentación del paquete, como manuales, información referente a códigos fuentes y archivos de configuración, la misma que se encuentra en diferentes formatos como txt, pdf, html, info y texi; de manera genérica, estos archivos, se los puede encontrar en /root/openss7-0.9.2.G/<nombre_paquete>/doc/manual/.

3.8 Operación del sistema instalado

Una vez instalados todos los paquetes de software de OpenSS7 detallados en el apartado anterior, se tiene un servidor Linux *stand-alone* con aplicaciones SS7 instaladas.

Los siguientes pasos para poder observar la operación del sistema instalado se reduce a:

- Identificación de los archivos de configuración de cada nivel de la pila SS7/SIGTRAN/VoIP.
- Determinar la nomenclatura de configuración de los archivos de acuerdo a la sintaxis de la programación usada.
- Configuración de los archivos de cada uno de los niveles de señalización, de acuerdo a la estandarización de cada uno de los protocolos.
- Ejecución de pruebas de manera metódica en cada nivel y/o protocolo de señalización para comprobar progresivamente el correcto funcionamiento.
- Identificación de los parámetros necesarios para poder integrar el servidor OpenSS7 a una red telefónica y pueda ser usado de acuerdo a la aplicación instalada: ISUP, SCCP, MAP, SIP y demás.

Identificación de archivos de configuración

Los archivos de configuración son:

- */etc/strchan.conf*
- */etc/strcompat.conf*
- */etc/streams.conf*
- */etc/strinet.conf*
- */etc/strisdn.conf*
- */etc/striso.conf*
- */etc/strload.conf*
- */etc/strsctp.conf*
- */etc/strsetup.conf*
- */etc/strsock.conf*
- */etc/strss7.conf*
- */etc/strtty.conf*
- */etc/strvoip.conf*
- */etc/strx25.conf*
- */etc/strxnet.conf*
- */etc/strxns.conf*
- */etc/sigtran.conf*

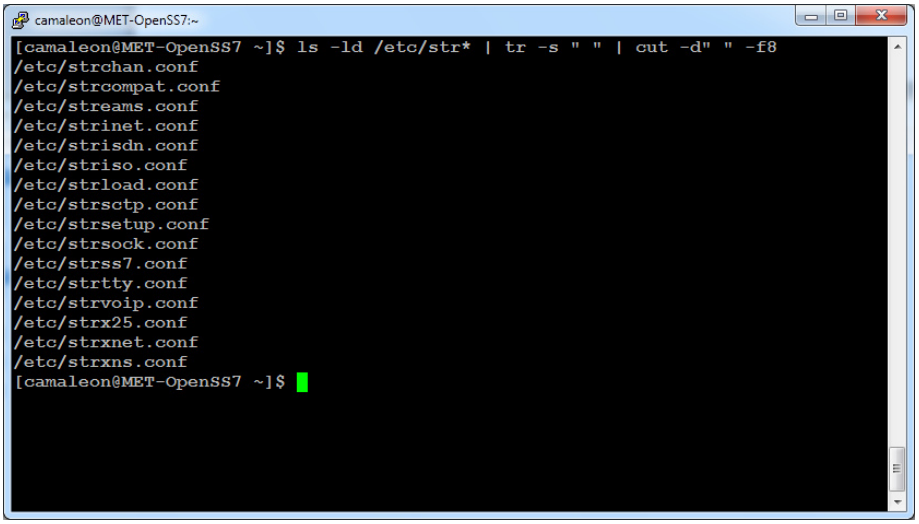


Figura 3.12. Archivos planos de configuración

Como se puede notar, se trata de archivos de configuración del tipo plano que se encuentran debajo del directorio */etc*, las configuraciones que aquí se detallan permiten que el aplicativo instalado pueda interactuar correctamente como tal, sin embargo existen otros archivos de configuración de estos mismos paquetes debajo de */etc/sysconfig* y */etc/rc.d/init.d/* que se refieren a las configuraciones necesarias para que los servicios y/o aplicaciones operen de manera correcta con el sistema operativo anfitrión, para las cuales se recomienda no modificar o ser muy cuidadoso en la manipulación.

Bajo el directorio */etc/rc.d/init.d/* se especifica los scripts o directorios de scripts que arrancan o detienen los servicios, mientras que en */etc/sysconfig* se encuentran los archivos de configuración de esos servicios que proporcionan los programas instalados.

Nomenclatura de configuración

Bajo condiciones normales, para cada equipo o sistema de telefonía que maneja señalización, que es adquirido y está en proceso de integración a la red de la operadora, sus archivos de configuración se encuentran en una condición tal, que permite que los valores de los parámetros de señalización o de red pueden ser remplazados por valores acorde a las necesidades de esa red o simplemente de mantener los valores por defecto.

Para el caso de los archivos de configuración de los paquetes instalados en el servidor OpenSS7 implementado, no todos vienen pre-configurados, es decir existe el archivo pero dentro del mismo solo se puede encontrar comentarios acerca del mismo, sin hacer referencia a los campos y/o parámetros a editar. A continuación una muestra de 2 archivos de configuración, Figura 3.13.

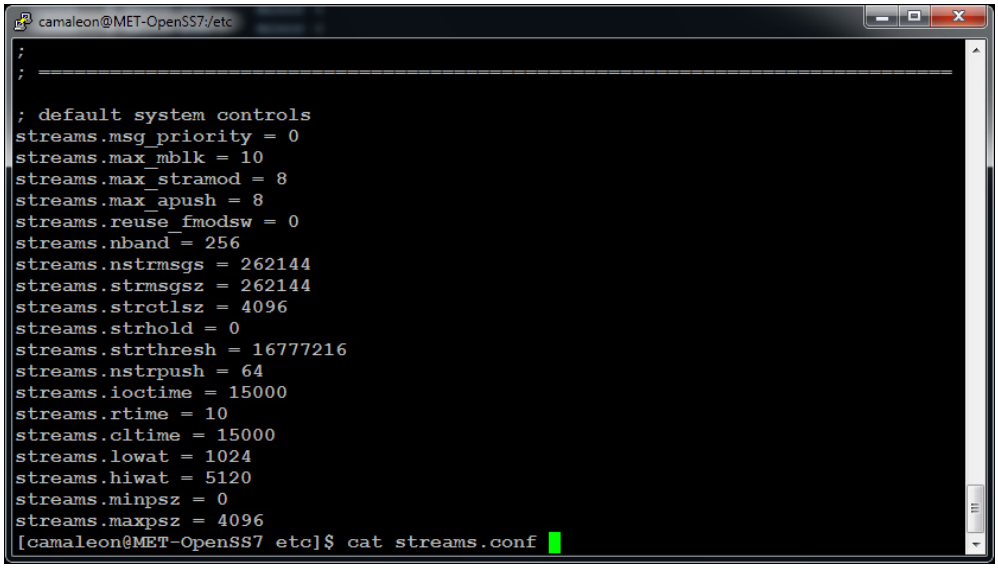


Figura 3.13. Contenido del archivo streams.conf

En la Figura 3.13 podemos apreciar que existen valores definidos para los distintos parámetros de STREAMS, lo cual debe ser evaluado en cuanto la conservación o el remplazo de dichos valores para una óptima operación.

Otro archivo interesante, es el *strsctp.conf*, en el cual se debe realizar las especificaciones de los parámetros del protocolo de transporte SCTP (Figura 3.14).

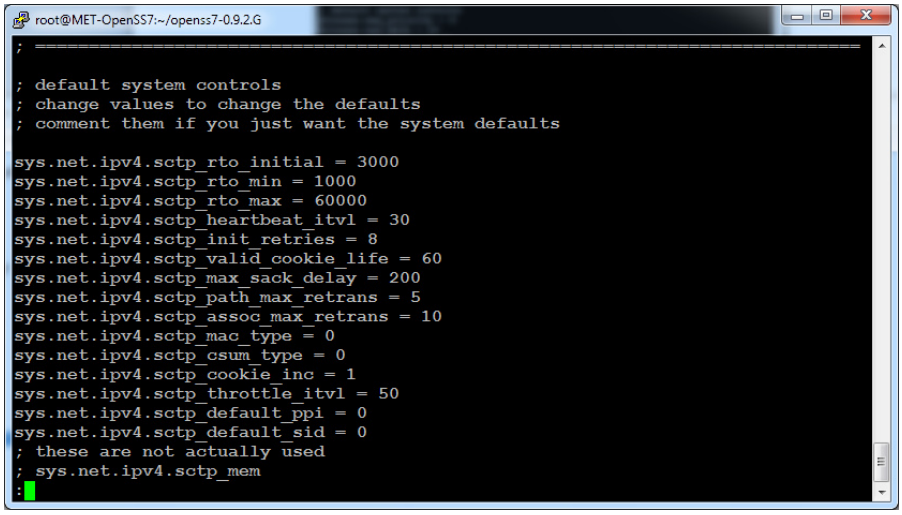
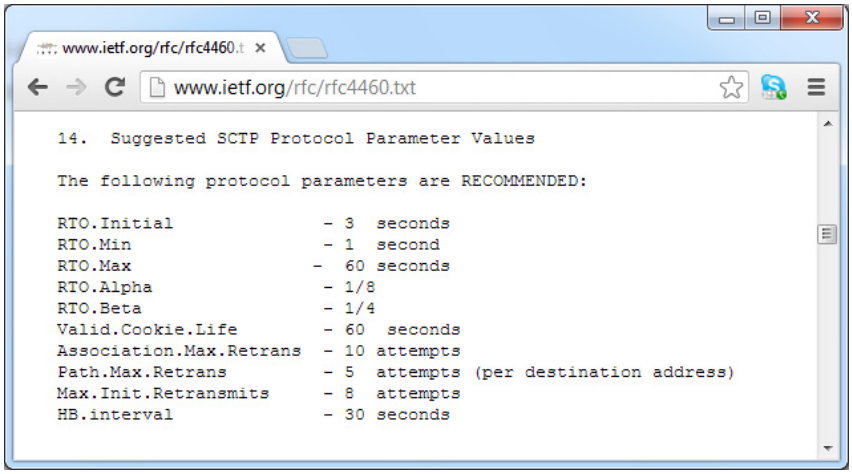


Figura 3.14. Valores de los parámetros de configuración de SCTP

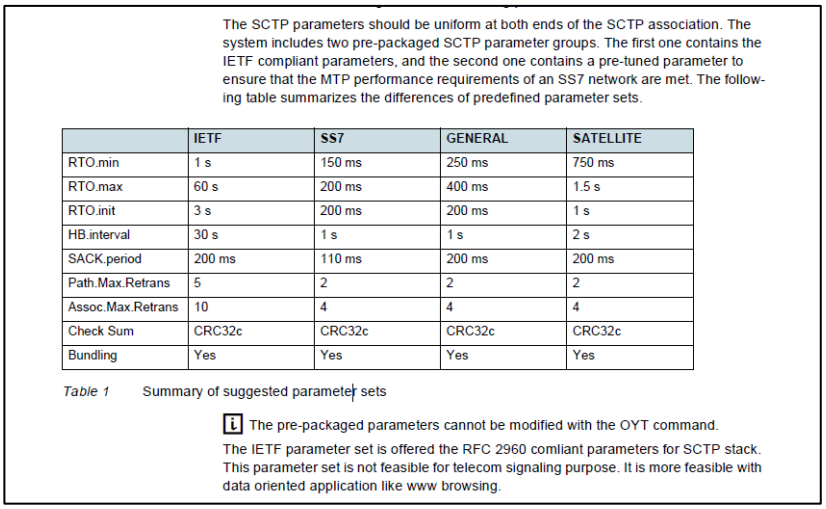
En la Figura3.14 se puede apreciar valores para los parámetros como RTO_INI, RTO_MIN, RTO_MAX, entre otros, los mismos que se ajustan a la sugerencia del RFC4460, cabe destacar que los valores de tiempo están proporcionados en mili-segundos (Figura 3.15).



Fuente: http://ietf.org/rfc/rfc4460.txt

Figura 3.15. Valores recomendados para SCTP según RFC4460

Los valores expuestos en la Figura 3.15 son los recomendados para SCTP en Internet, sin embargo para redes privadas de las operadoras, que es donde operan los servidores de señalización, los valores debe ajustarse como lo mostrado en la Figura 3.16.



The SCTP parameters should be uniform at both ends of the SCTP association. The system includes two pre-packaged SCTP parameter groups. The first one contains the IETF compliant parameters, and the second one contains a pre-tuned parameter to ensure that the MTP performance requirements of an SS7 network are met. The following table summarizes the differences of predefined parameter sets.

	IETF	SS7	GENERAL	SATELLITE
RTO.min	1 s	150 ms	250 ms	750 ms
RTO.max	60 s	200 ms	400 ms	1.5 s
RTO.init	3 s	200 ms	200 ms	1 s
HB.interval	30 s	1 s	1 s	2 s
SACK.period	200 ms	110 ms	200 ms	200 ms
Path.Max.Retrans	5	2	2	2
Assoc.Max.Retrans	10	4	4	4
Check Sum	CRC32c	CRC32c	CRC32c	CRC32c
Bundling	Yes	Yes	Yes	Yes

Table 1 Summary of suggested parameter sets

The pre-packaged parameters cannot be modified with the OYT command.

The IETF parameter set is offered the RFC 2960 comliant parameters for SCTP stack. This parameter set is not feasible for telecom signaling purpose. It is more feasible with data oriented application like www browsing.

Fuente: Nokia Siemens Network

Figura 3.16. Valores de SCTP clasificados por ambiente según Nokia Siemens Networks

Desafortunadamente, no todos los archivos de configuración vienen con valores pre-definidos, existen otros como *sigtran.conf* que no contiene ni el formato ni menciona los parámetros a los que se les debe asignar los

valores, solo se encuentran comentarios acerca los antecedentes y origen del archivo, Figura 3.17.

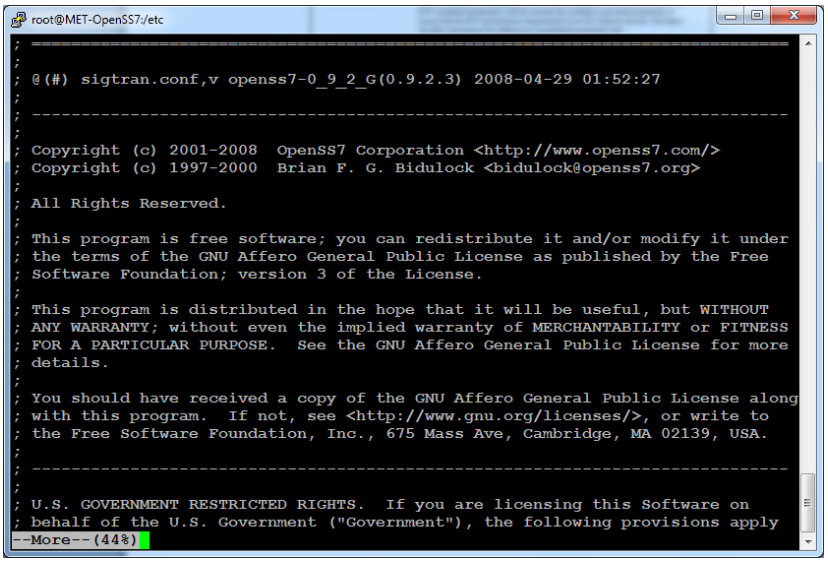


Figura 3.17. Contenido del archivo sigtran.conf

Por este motivo, es necesario realizar una investigación más profunda acerca del formato y nomenclatura a usar en este tipo de archivos para su configuración, la guía para lograr este cometido se encuentra en los manuales, archivos cabeceras (*.h*) y archivos de programación (*.c*) de cada Paquete a instalar:

- Manuales
  - */root/openss7-0.9.2.G/<nombre_paquete>/doc/manual*
- Archivos cabecera y de programación
  - */root/openss7-0.9.2.G/<nombre_paquete>/src/include*

Configuración adecuada de los valores

Una vez determinado el formato adecuado de ingresar los parámetros dentro de los archivos de configuración, se debe ingresar de manera apropiada los valores para éstos parámetros, para ello se recomienda ajustarse a las estandarizaciones de los protocolos. Por ejemplo, para la parte de SIGTRAN se debe definir el nivel M3UA con los parámetros mostrados en la Figura 3.18.

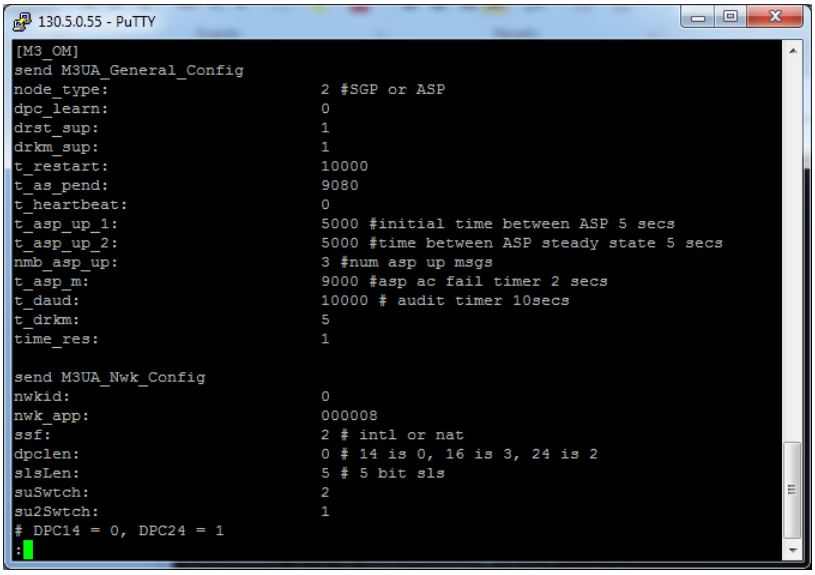


Figura 3.18. Muestra de configuración M3UA en un IVR Tecnotree

Pruebas por niveles

Para ciertos paquetes, existe una serie de pruebas que se pueden ejecutar para comprobar su correcta funcionalidad e instalación. Estas pruebas en su

mayoría están recopiladas en un ejecutable llamado *testsuite* como se muestra en Figura 3.19.

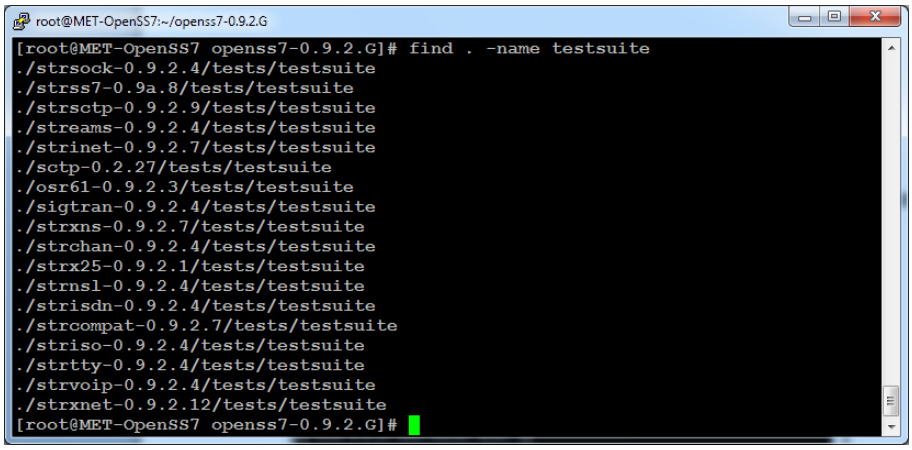


Figura 3.19. Pruebas a ejecutar por Paquete a instalar

Durante la ejecución de las pruebas se puede observar (Figura 3.20).

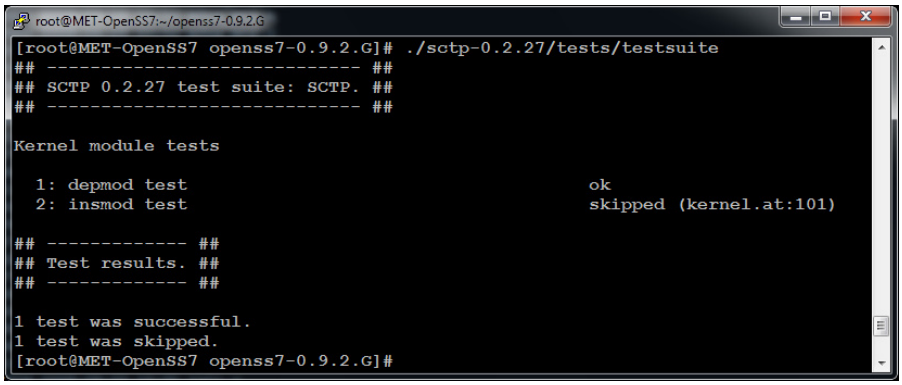


Figura 3.20. Ejecución de *testsuite* para sctp-0.2.27

Luego de la fase de pruebas se genera un log dentro del directorio de pruebas, organizado de acuerdo al número de prueba ejecutada, por ejemplo:

- El log./sigtran-0.9.2.4/src/drivers/testsuite.dir/235/testsuite.log, indica que se generó un archivo tipo log para la prueba 235 del testsuite del paquete sigtran-0.9.2.4.

- Mayor información acerca del manejo de las pruebas se puede encontrar en los manuales de los paquetes instalados.

Virtual Machine

Otra alternativa, muy válida, para realizar la implementación del servidor OpenSS7, es hacerlo en una máquina virtual, lo cual fue realizado para el presente trabajo.

Se escogió *Oracle VM Virtual Box* como la herramienta de virtualización para montar un sistema operativo Fedora Core 9 sobre un sistema operativo Microsoft Windows 7. A continuación se puede apreciar las características de la máquina virtual Linux configurada (Figura 3.21).

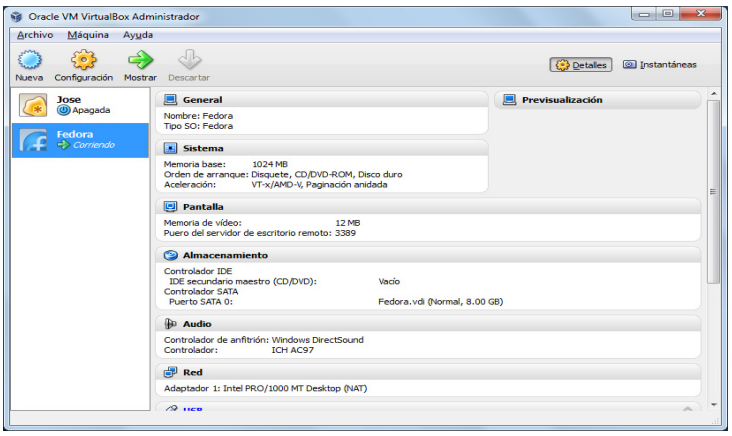


Figura 3.21. Máquina Virtual Fedora Core 9

Una vez configurada de manera adecuada la máquina virtual, ésta puede operar de igual manera que una real. En la mencionada se puede ver como los servicios de STREAMS (paquete de OpenSS7) son levantados al arrancar el sistema (Figura 3.22).

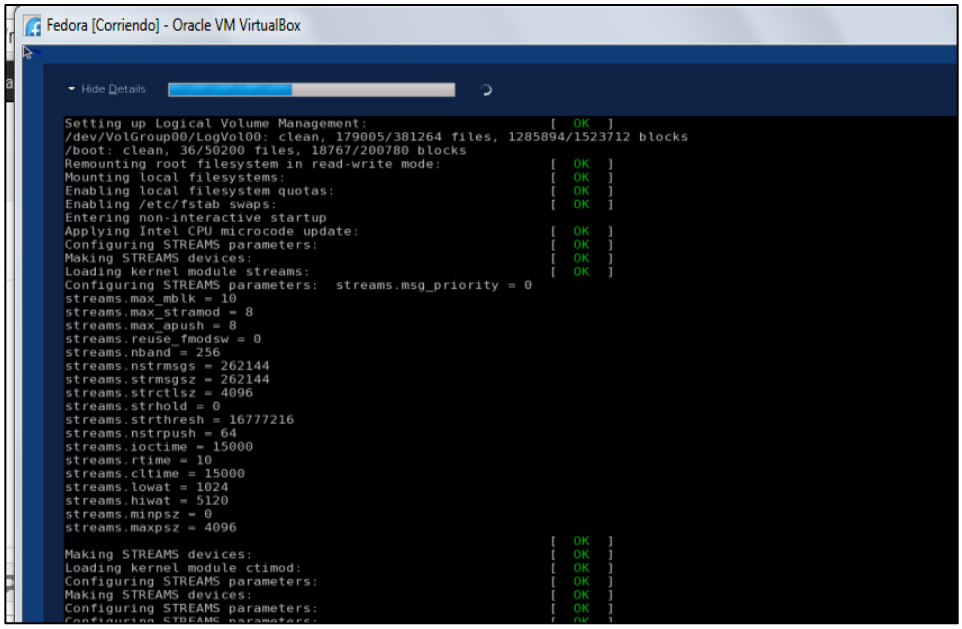


Figura 3.22. Estado de los servicios al arrancar Fedora Virtual

Inclusive, los paquetes de OpenSS7 en su totalidad pueden ser instalados sin representar algún trato diferente, como se lo puede denotar en la Figura 3.23.

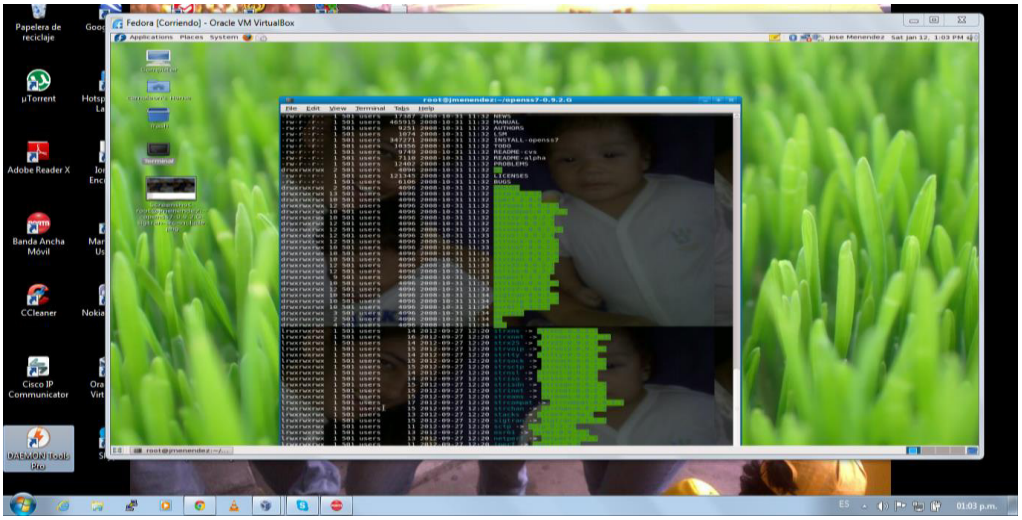


Figura 3.23. Paquetes OpenSS7 instalados en la máquina virtual

3.9 Posibles aplicaciones

Teniendo como partida un servidor Linux con aplicaciones SS7/SIGTRAN/VoIP instaladas, es suficiente poder inferir que el complementar con ciertos aspectos técnicos adicionales, transforma este trabajo como la base para el desarrollo de otras plataformas, como por ejemplo:

- El enfocar las configuraciones del servidor a ejecutar protocolos como SCCP sumado el protocolo CAMEL como la aplicación que opere sobre SCCP y una base de datos como Sybase, se tiene una pequeña plataforma: Prepago (Anexo A) o un HLR, este modelo señalado es fácilmente implementado en el ambiente de una red de un operador móvil.

- Si las configuraciones son perfiladas a ejecutar los protocolos MAP y SMPP se puede transformar en una plataforma de valor agregado muy difundida y sobretodo valorada por las operadoras celulares: un SMSC con la particularidad de que no podría ejercer las funciones de *Store & Forward* (es decir una plataforma que no guarde los SMS's para luego reintentar la entrega en caso de no poder hacer lo en el primer intento).
- Hacer las funciones de PTS, un concentrador de señalización de una red telefónica.

Todos éstos son ejemplos de mucha relevancia en una red telefónica y por ende para un operador telefónico hoy en día.

CAPÍTULO 4

4 ESTADO DEL PROYECTO OPENSS7

El proyecto OpenSS7, debido a la amplia cobertura en temas de señalización SS7, sigue desarrollando componentes y liberando etapas por ejes de acción, actualmente se encuentra en estado de producción, lo que no quiere decir que aún no existan liberaciones de paquetes de software al público, luego de que estos han superado con éxito las diferentes fases para ello.

4.1 Avances realizados

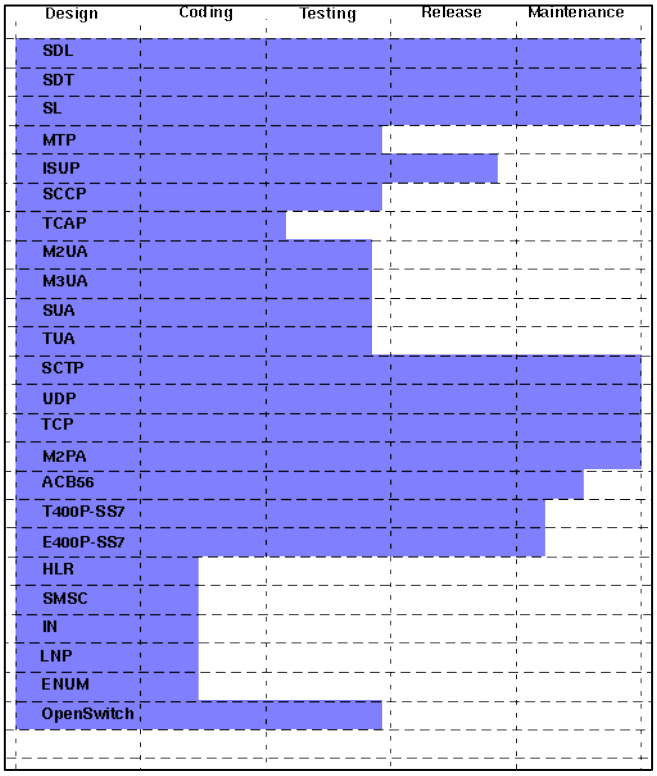
Existen componentes que se encuentran es estado Alfa o Beta (últimos estados de desarrollo), sin embargo existen otros que ya están disponibles para la difusión pública, pero lo más importante de todo esto es que existe la

liberación del código fuente de los paquetes de software, para que el que esté interesado pueda contribuir con el desarrollo ya logrado y realice avances en el tema.

De la misma manera, OpenSS7 Corporation, sostiene que pese a la liberación al público de los componentes estables en operación, no significa que la carga y manipulación del código fuente no represente un riesgo para el sistema operativo anfitrión, ya que puede bloquear la máquina y tener luego que reiniciarla, o inclusive podría dañar los archivos cabeceras y no poder volver a compilar el kernel del sistema operativo. Por ello el estado en producción del proyecto.

Una de las facilidades que ofrece OpenSS7 Corporation, es dar acceso a los archivos tipo *Valores Separados por Coma (Comma-Separated Values, CVS)* a quienes se registren bajo el perfil de subscriptores y sponsors, y para tal categoría se debe cumplir con mínimos requisitos de acuerdo al formulario a llenar en la web.

A continuación se puede apreciar el estado en que se encuentran los diferentes componentes OpenSS7 (Figura 4.1).



Fuente: http://www.openss7.org/roadmap.html

Figura 4.1. Avance de los componentes y aplicaciones de pila del proyecto

4.2 Sistema Operativo usado

El principal criterio del proyecto OpenSS7 para usar a Linux como el sistema operativo anfitrión, es el pertenecer a la familia de sistemas abiertos con licencia LPG GNU al igual que OpenSS7, lo que también conlleva al hecho de que la interacción entre el núcleo o kernel y los programas y aplicaciones al usuario tengan la misma característica de licencia.

Adicionalmente, la arquitectura de Linux se presta para realizar las adecuaciones necesarias para la instalación y desarrollo de aplicaciones de

manera modular como en el presente caso implementación de OpenSS7, en la Figura 4.2 se muestra esta modularidad.

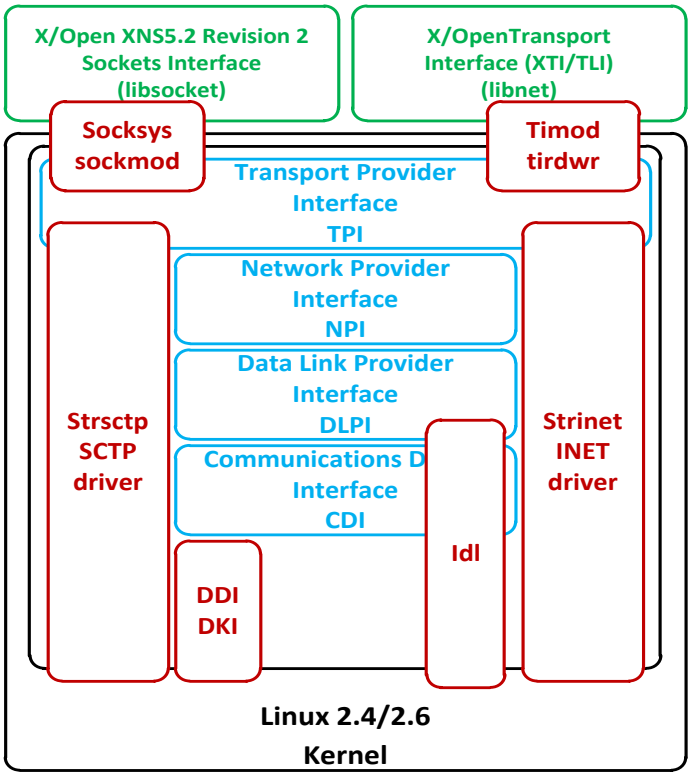


Figura 4.2. Estructura de implementación de OpenSS7 sobre Linux

Como se puede apreciar, existen ciertos módulos sobre el kernel 2.4 o 2.6 de Linux a integrar, cuyas funciones se pasan a detallar [36]:

- *Enlace de Datos Linux (Linux Data Link, LDL)*: este controlador provee una interfaz entre los controladores STREAMS y los controladores de red

- *Controlador INET STREAMS*: es una colección de controladores y módulos, que proporciona capacidades INET en una serie de formas relacionadas, ya sean TCP, UDP y variantes de IP.
- *Controlador SCTP STREAMS*: controlador del protocolo SCTP, el cual ha sido implementado como un STREAM.
- *Módulo Compatibilidad STREAMS*: este es el paquete strcompat-0.9.2.7 mencionado en el capítulo 3.
- *Linux Fast-STREAMS*: es el punto fundamental del proyecto OpenSS7, STREAMS.
- *Interfaz de Transporte X/Open (X/Open Transport Interface, XTI)*: se trata de la lilbrería XTI y XNET, libxnet.
- *Interfaz de Proveedor de Transporte (Transport Provider Interface, TPI)*: define una interfaz de mensaje para el proveedor de transporte implementado en STREAMS. Así, un usuario se comunica con un proveedor de transporte a través de un camino full-dúplex conocido como un *stream*. Este *stream* proporciona un mecanismo en el que los mensajes se pueden pasar al proveedor de transporte desde el usuario del transporte y viceversa.
- *Interfaz de Proveedor de Red (Network Provider Interface, NPI)*: define los servicios proporcionados por la capa de red al usuario de red, en su frontera.

- *Interfaz de Proveedor de Enlace de Datos (Data Link Provider Interface*, *DLPI)*: es una implementación del estándar que define los servicios proporcionados por la capa de enlace Datos del modelo ISO y especifica una interfaz con los mismos.
- *Interface de Dispositivo de Comunicación (Communications Device Interface, CDI)*: es una interfaz de dispositivo de bajo nivel que se define entre un usuario y un proveedor con STREAMS y el sistema de llamadas putmsg y getmsg.
- *Interfaz de Red de Área Amplia (Wide Area Network Interface, WAN)*: es una interfaz de servicio *primitive* para la capa de controlador del controlador WAN.

4.3 Sistemas que operan en el mercado con SS7 sobre Linux o Solaris

En el amplio Mundo de la telefonía y en particular de las telecomunicaciones, existen una variedad inmensa de equipos y soluciones orientadas a satisfacer las necesidades de los abonados a través de servicios implementados en estos equipos, la diversidad es muy amplia, sin embargo, el denominador común es que todas estas soluciones obedecen a estandarizaciones de protocolos para comunicación, operación y mantenimiento; pese a esto, los fabricantes y propietarios comercializan sus productos con diferencias claras que les permiten tomar ventajas sobre su

competencia. Es así que, empresas como Life Tree Corporation han implementado sus aplicaciones SCCP sobre un sistema operativo Linux, donde el servicio corresponde a la interfaz de señalización a una *Red Inteligente* o también conocida como Prepago (Figura 4.3).

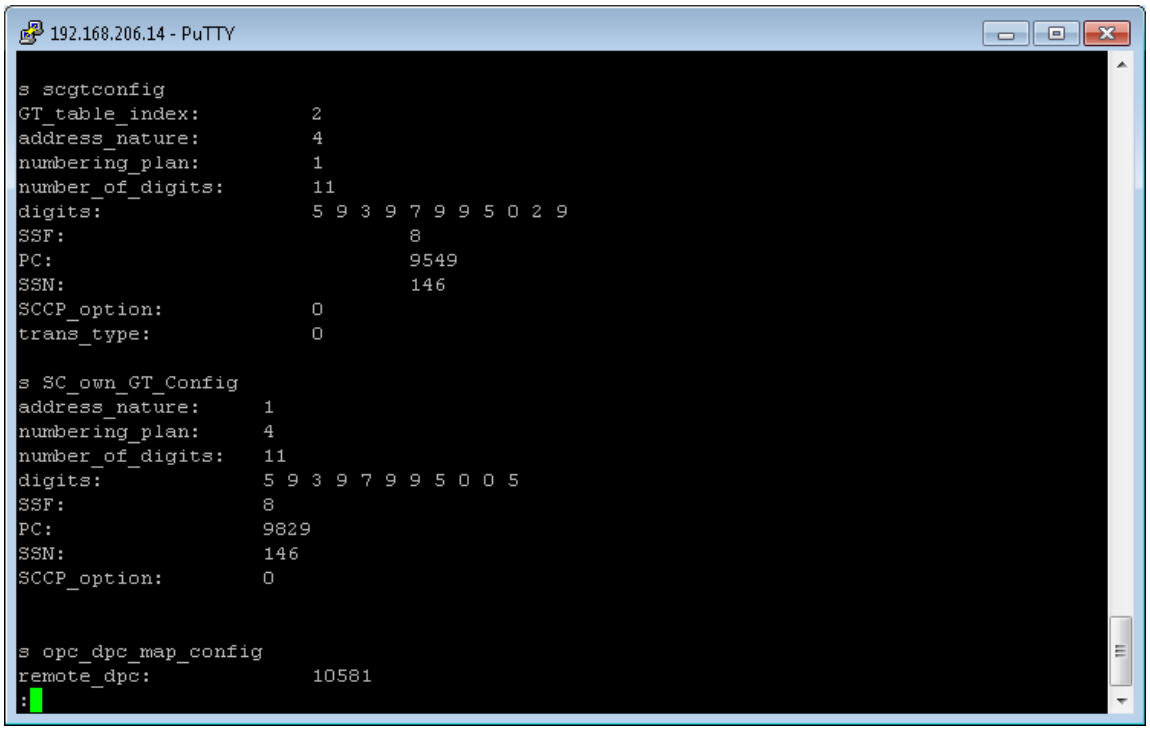
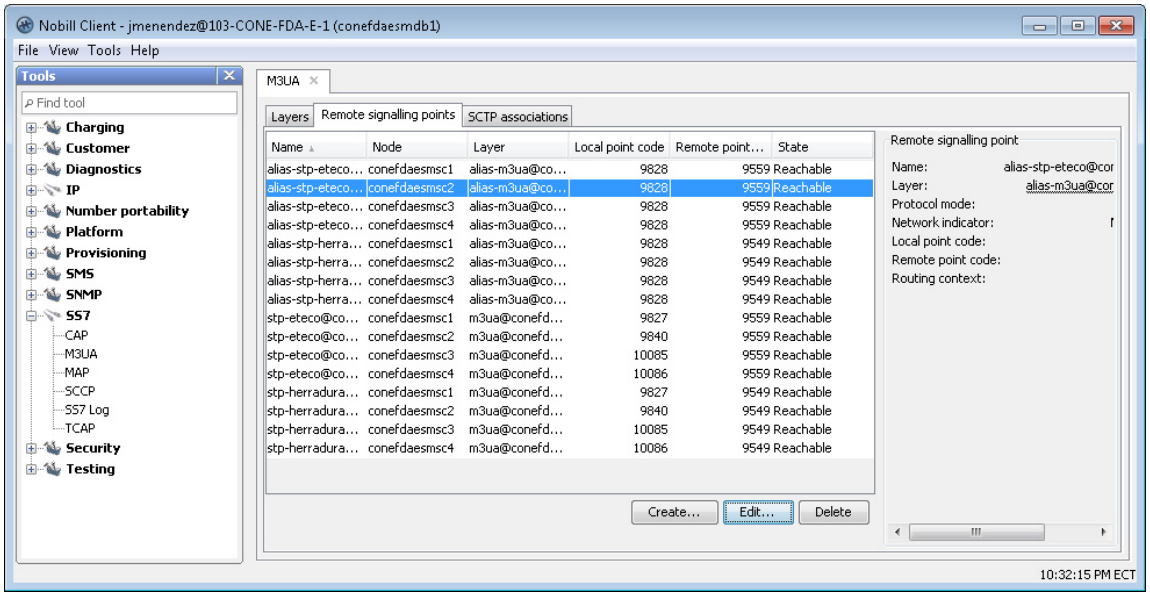


Figura 4.3. Archivo plano de configuración de protocolo SCCP

Otro fabricante que ha optado por hacer funcionar su plataforma de SMSC con señalización sobre un Solaris, teniendo una interfaz gráfica desarrollada en Java para realizar el monitoreo y configuraciones, es el sueco Symsoft, a continuación una ventana principal que muestra las configuraciones realizadas para la capa de M3UA (Figura 4.4).



Fuente: Aplicativo Nobill de Symsoft

Figura 4.4. Interfaz gráfica para el monitoreo y configuración de SS7

Así como estos dos gigantes europeos, existen un sinnúmero de fabricantes que tiene como preferencia desarrollar sus productos en un sistema operativo Linux y derivados de este, como la solución *Nastar* de Huawei, *Visual Voicemail* de Comverse, la *PBX Crystal* de Alcatel, y otros.

4.4 Ventajas de uso de sistemas abiertos

Sin lugar a duda, el desarrollo de los sistemas abiertos ha tomado mucho cuerpo en los últimos años, y la adopción de éstos sistemas por partes de operadoras telefónicas, instituciones gubernamentales, educativas, industria... Se funda principalmente por:

- El costo: la ausencia del rubro dedicado a la licencia de un sistema es un significativo ahorro para las operadoras, mientras que para los abonados finales también existen rubros que desaparecen como es el caso del servicio de VoIP, que permite adicionar funcionalidades telefónicas que para las líneas regulares representaría un costo adicional.

- Las mejoras desarrolladas heredan la condición de sistemas abiertos: cualquier cambio o mejora que implique una redistribución nueva, debe tener la característica de software libre, que permitirá el beneficio de la comunidad.

- Seguridad: el hecho de tener acceso al código fuente, permite a los interesados en seguridad a nivel mundial, que detecten posibles errores y desarrollen parches para mitigar vulnerabilidades, así se mantiene una mejora continua, un ejemplo es el caso de los protocolos de seguridad sobre IP para el servicio de VoIP como lo son IPSEC y TLS.

- Independencia: el no estar atado a una licencia, permite a las empresas actuar de manera independiente respecto a los proveedores de software.

- Interoperabilidad: el seguir y cumplir fielmente con las estandarizaciones dentro de los sistemas abiertos permite que estos sean altamente interoperables.

- Nuevos desarrollos: en los sistemas abiertos correspondientes a la telefonía es fácil caer en cuenta que existen mejoras en los servicios como es el caso de la portabilidad, en VoIP se puede tener el servicio en una línea instalada en una PC, en un terminal de escritorio o inclusive en un terminal móvil, así mismo, el desarrollo de mejoras de nuevos *codecs* mejora sustancialmente la calidad de audio llegando en ocasiones a superar al de la telefonía tradicional, por lo tanto, es fácil realizar nuevos desarrollos ya sea para nuevas aplicaciones o mejoras en el sistema anterior, esto debido a la disponibilidad y apertura del código fuente.

4.5 Futuro de aplicaciones comerciales sobre plataformas open-source

En la actualidad existen muchas soluciones que operan sobre plataformas *Código Abierto*, no solo en el campo de la VoIP sino también en otras áreas tecnológicas, por contar con grandes ventajas frente a los sistemas propietarios, como se vio en el punto anterior. Sin embargo, en ocasiones, la independencia que se goza al no tener que adquirir una licencia, se ve un tanto sesgada al momento de dar el mantenimiento, es el caso, por ejemplo, de Red Hat Enterprise Linux, RHEL, distribución de Linux que como otro derivado de GNU/Linux es completamente gratis y con *Código Abierto*, pero no así su mantenimiento, para el cual se debe firmar un contrato con RHEL,

quienes se encargan de realizar las actualizaciones y corregir cualquier error encontrado en la versión adquirida.

Por otro lado, si se escoge un sistema operativo de *Código Abierto* que no disponga de mantenimiento (como podría ser el caso de algunas versiones de Ubuntu), también se produciría un problema en el caso que se produzcan fallos en el kernel.

Por lo tanto, el presente y el futuro de las aplicaciones comerciales funcionando sobre sistemas abiertos, dependen de las decisiones de las compañías de atar su solución a una licencia y a un contrato de mantenimiento para el caso de aplicaciones propietarias, o tan solo a firmar un contrato de mantenimiento para los sistemas abiertos.

CAPÍTULO 5

5 ANÁLISIS DE RESULTADOS

5.1 Resultados obtenidos

Al finalizar la implementación, se obtiene como resultado un servidor Linux con kernel 2.6 *stand-alone* capaz de hacer funcionar aplicaciones del SS7 con una adecuada configuración para la pilas de SCTP, SIGTRAN, VoIP y SS7 TDM.

La configuración se logra a través de archivos planos que hacen referencia a cada componente de señalización instalado y ubicados debajo del directorio /etc/, sin embargo, para lograrlo, primero se debe determinar el formato y nomenclatura a usar en cada archivo, a más de tener un concepto amplio de los parámetros y sus valores a colocar por cada componente.

La misma implementación es posible lograr en un ambiente de máquina virtual, usando cualquier herramienta de virtualización encontrada en la web.

Los paquetes instalados vienen con un set de pruebas con el fin de comprobar no solo la correcta instalación de los paquetes sino también de la correcta operación del componente de señalización que ese paquete representa, sin embargo la mayoría de pruebas están relacionadas a evaluar la interacción a nivel de red con otros *signaling points* o *signaling gateways*, en otras palabras se requiere de al menos un servidor OpenSS7 adicional, ambos con una adecuada configuración a nivel de SCTP según lo requerido en el archivo */etc/strsctp.conf* de cada servidor (configuración que no es cubierta por el alcance del presente trabajo, y principal motivo para no poder realizar la integración entre el servidor *desktop* y la máquina virtual), para así poder completar el *set* de pruebas.

5.2 Interpretación de Resultados

Como ya se había mencionado en el apartado 3.8, el proyecto OpenSS7, instala también un set de pruebas especializadas por cada paquete instalado, el mismo que serviría para probar cada uno de los aspectos y adecuadas configuración inherentes a la operación del componente que se ha instalado en el sistema.

El grupo de pruebas puede ser accionado ejecutando un script que recopila a todas pruebas competentes para ese paquete, y es el: *testsuite.*

Para el caso del paquete *strsctp-0.9.2.9*, que provee todos los requisitos y programación necesaria para que opere el protocolo de transporte SCTP e interactúe con el resto del sistema, el conjunto de pruebas se encuentra en el directorio */root/openss7-0.9.2.G/strsctp-0.9.2.9/tests*, en este directorio se puede encontrar los scripts individuales:

- *test-sctp_t*
- *test-sctp_n*
- *test-sctp_n2*

Los cuales están programados en lenguaje C y que contienen pruebas respecto a los drivers de SCTP, para las TPI, NPI y neo-NPI, respectivamente. Así mismo en el mismo directorio ya mencionado encontramos a *testsuite*, que permite con el cual se tiene una mayor validación y además ejecuta las pruebas contenidas en los scripts previamente señalados, la Figura 5.1 muestra los archivos en cuestión.

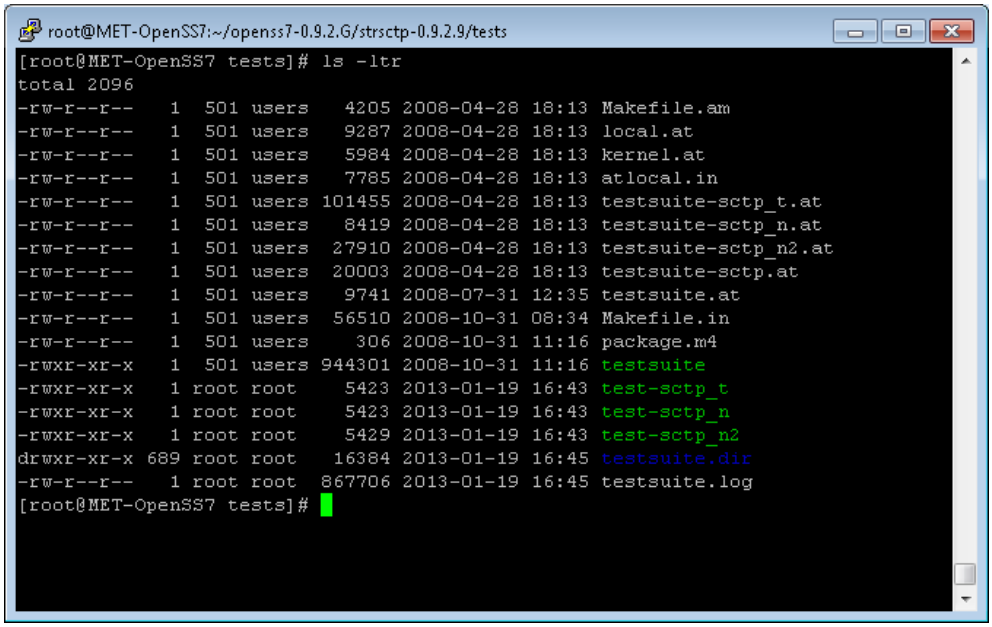


Figura 5.1. Archivos dentro del directorio /root/openss7-0.9.2.G/strsctp-0.9.2.9/tests

Para poder ejecutar las pruebas de debe ejecutar el scrip testsuite mediante: *./testsuite* dentro del directorio: */root/openss7-0.9.2.G/strsctp-0.9.2.9/tests*. Una vez ejecutado el script, se puede observar el avance de las pruebas como el resultado de las mismas, como se muestra en la Figura 5.2 luego de haber ejecutado la orden *./testsuite*.

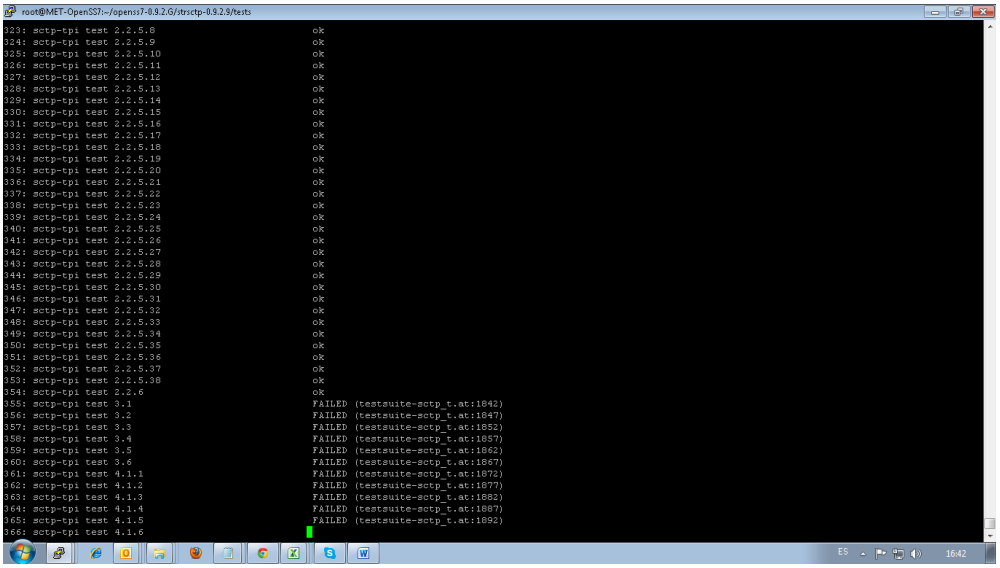


Figura 5.2. Ejecución de pruebas de controladores SCTP

La ejecución toma alrededor de un minuto o menos, luego de esto, se crea un archivo *testsuite.log* dentro del directorio */root/openss7-0.9.2.G/strsctp-0.9.2.9/tests* donde se registran todas las novedades de la ejecución, en la parte final del contenido del archivo, se detallan las pruebas que no fueron exitosas y se especifica como enviar, por correo, al grupo OpenSS7 para reportar tales novedades, además de incentivar claramente a que la investigación se haga por cuenta propia (Figura 5.3).

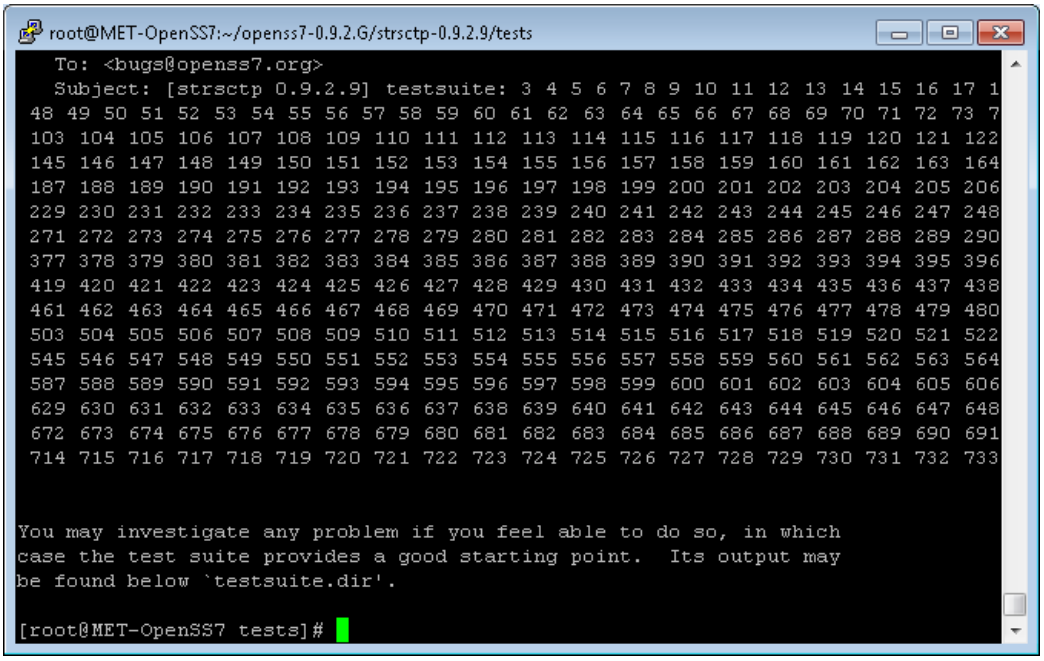


Figura 5.3. Contenido del archivo testsuite.log

Adicionalmente a esto también se crea un directorio llamado *testsuite.dir* dentro del cual se puede ver que existe un subdirectorio por cada número de prueba fallada, las mismas que describe el archivo *testsuite.log* (Figura 5.4).

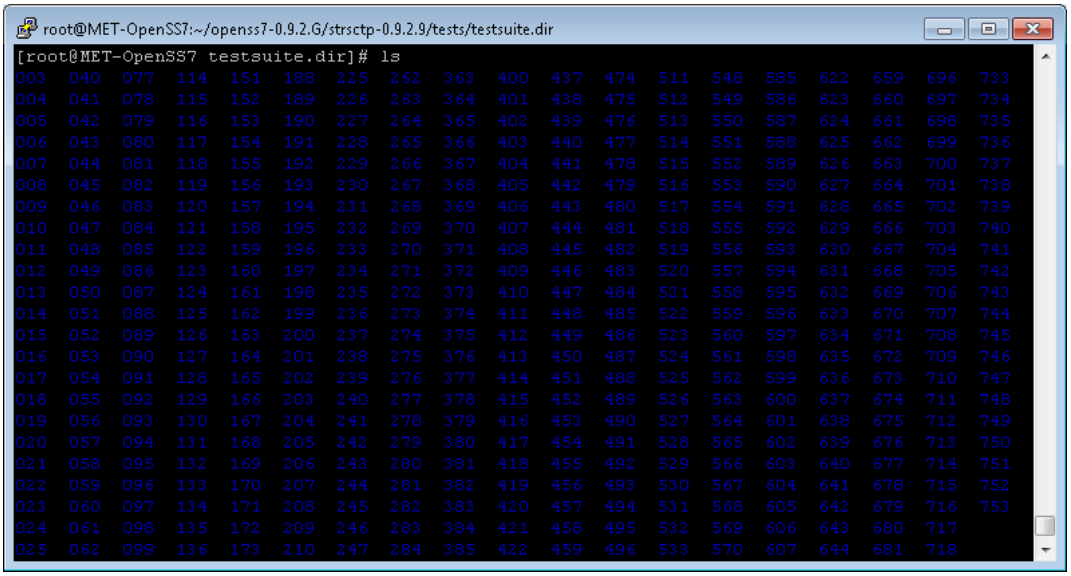


Figura 5.4. Contenido del directorio testsuite.dir

Dentro de cada uno de los subdirectorios, los cuales están nombrados con números, se puede hallar un log individual llamado: *testsuite.log*, el cual pertenece a la prueba ejecutada según el nombre del directorio. Es necesario indicar que el resultado de cada una de las pruebas puede tomar uno de los siguientes valores: PASS, FAIL, PASS, NOT APPLICABLE, INCONCLUSIVE o SKIPPED.

A manera de ejemplificar lo mencionado, a continuación, se lee el log del subdirectorio *444* donde claramente, primero, se especifica a que grupo de prueba pertenece y se señala el motivo por el cual no fue exitosa la prueba (Figura 5.5).

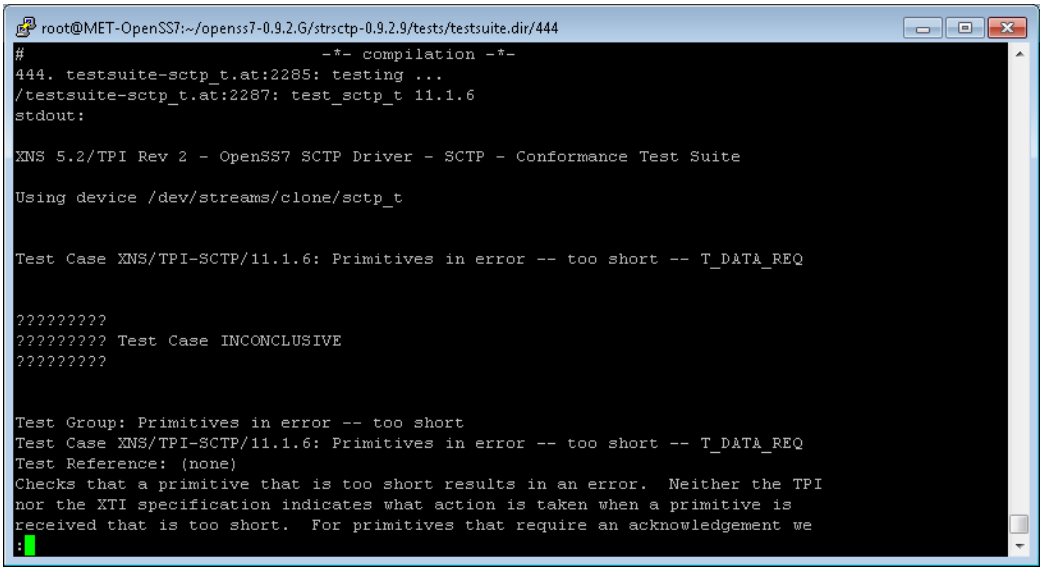


Figura 5.5. Resultado de la prueba 444 para las prueba ejecutadas para el paquete STRSCTP

La prueba fue calificada como INCONCLUSIVE, y el tópico a probar fue: *Primitives in error -- too short -- T_DATA_REQ* (que es la señalada en la especificación: *XNS/TPI-SCTP/11.1.6*), el escenario de esta prueba se centraba en enviar un *primitive* en error, alegando el hecho de ser muy corto, esto dentro del *STREAM* de la asociación establecida. La prueba es inconclusa debido a que el servidor donde se lo prueba está en modo *stand-alone* es decir no realiza ninguna interacción con otra entidad de señalización y por ende no tiene configurada ninguna asociación.

En la figura 5.6 se muestran las pruebas que, según el *testsuite.log* ubicado en */root/openss7-0.9.2.G/strsctp-0.9.2.9/tests*, fueron exitosas, con el tiempo

de ejecución, estas pruebas hacen referencia a la operación local de los controladores de los NPI y TPI del servidor OpenSS7.

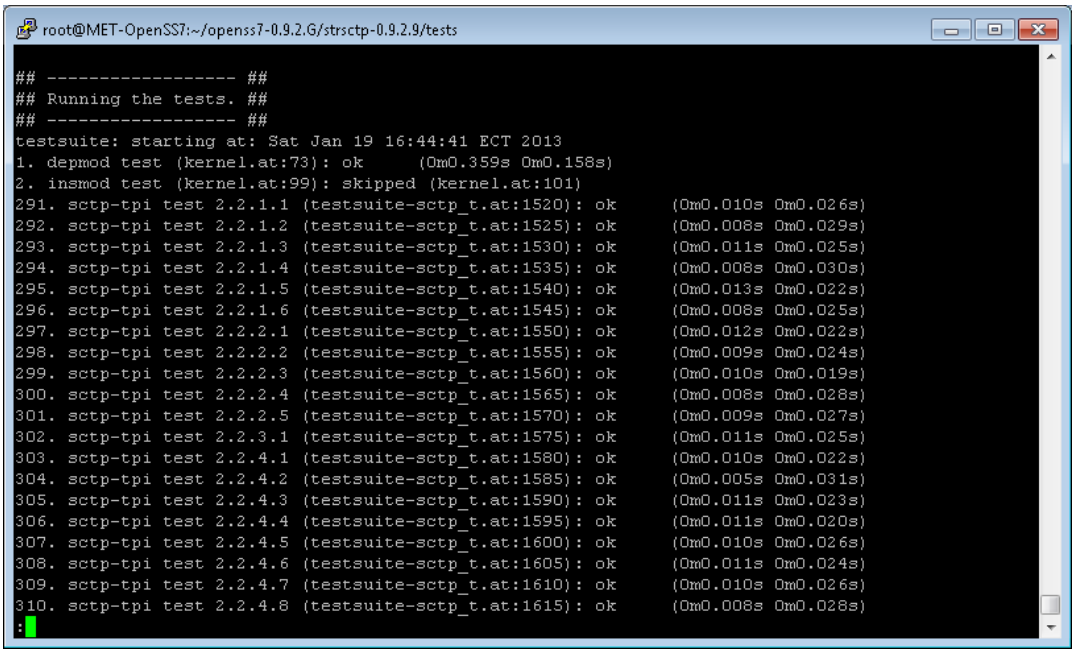


Figura 5.6. Pruebas calificadas como exitosas

5.3 Contraste económico

Para poder apreciar de manera adecuada la dimensión de las diferencias de los productos de licencia comercial versus las soluciones que se deben a una licencia *Código Abierto*, es conveniente realizar mediante la ejemplificación numérica.

Cabe recalcar que las soluciones de servicios y/o productos de telefonía no vienen por separados, así vemos que una solución de *casillero de voz* contemplan los módulos de software y hardware capaz de proveer la

señalización SS7, señalización ISUP, la interfaz de voz, el almacenaje para guardar los mensajes de voz, el gestor de monitoreo de la plataforma... Es decir, que para nuestro caso, la tarea de poder comparar nuestro servidor SS7 *stand-alone*, es algo complicada, sin embargo las propuestas comerciales siguientes pueden dar una idea del alcance monetario que puede llegar a tener una solución que opere bajo los servicios de SS7.

Escenario 1: plataforma *MMS-Gateway*, solución inter-operadora, capaz de pasar tráfico de SMS/MMS de una operadora a otra.

La especificación a nivel de SS7 se muestra en la Figura 5.7.

SS7

Interface	Standards
MTP	ITU-T, Q.701-70
SCCP	ITU-T Q.711-716
TCAP	ITU-T Q.771-775
SCTP	RFC 2960
SIGTRAN	M3UA RFC 3332
MAP version 1,2 and 3	ETS 300 974, GSM 09.02, 3GPP TS
CAP version 1, 2, 3 and 4	GSM 09.78, 3GPP TS 29.078

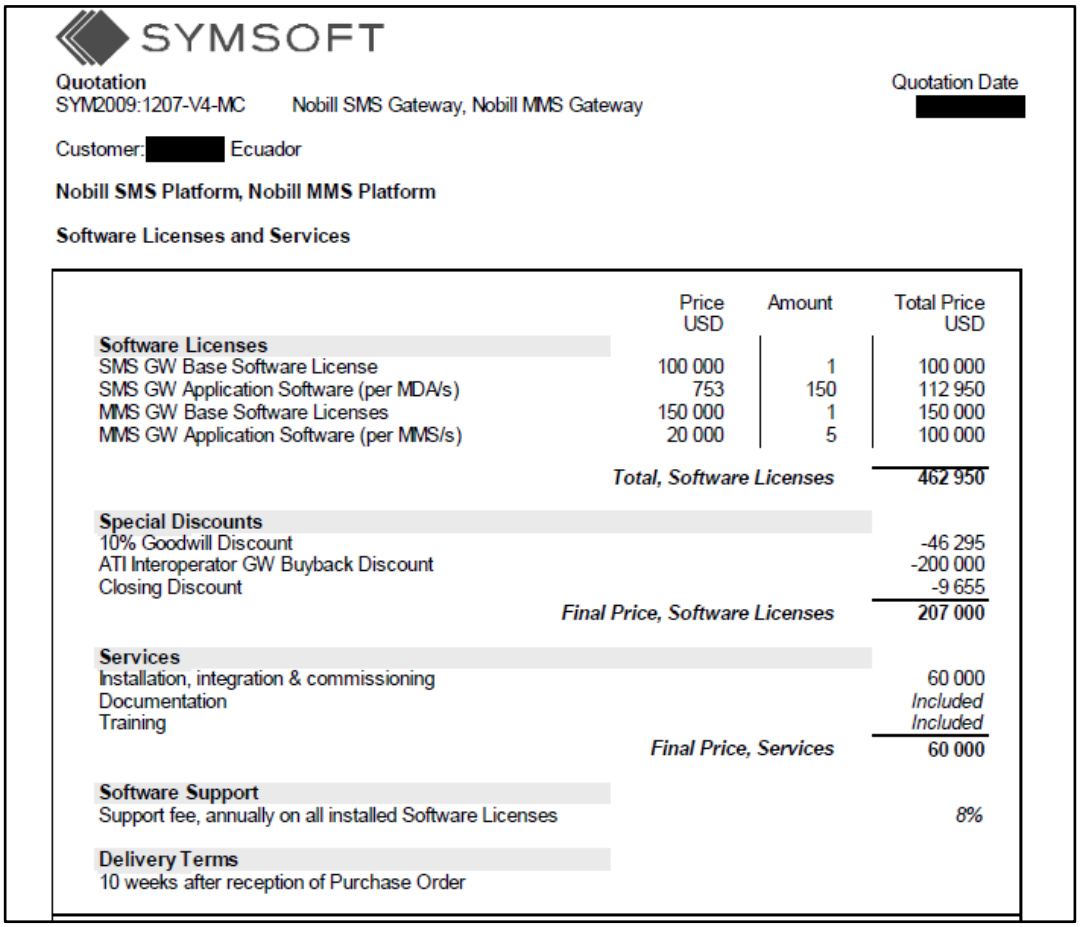
Fuente: Propuesta comercial digital enviada por Symsoft

Figura 5.7. Especificación SS7 en propuesta de MMS/SMS Gateway

Propuesta Symsoft

De acuerdo a la propuesta comercial enviada por el fabricante sueco, Symsoft, donde ofrecen una plataforma de alta disponibilidad y con una serie de características técnicas capaz de satisfacer las necesidades del operador

móvil, requeridas para pasar el tráfico de SMS y MMS a otras operadoras móviles (Figura 5.8).



SYMSOFT

Quotation
SYM2009:1207-V4-MC Nobill SMS Gateway, Nobill MMS Gateway

Quotation Date [redacted]

Customer: [redacted] Ecuador

Nobill SMS Platform, Nobill MMS Platform

Software Licenses and Services

	Price USD	Amount	Total Price USD
Software Licenses			
SMS GW Base Software License	100 000	1	100 000
SMS GW Application Software (per MDA/s)	753	150	112 950
MMS GW Base Software Licenses	150 000	1	150 000
MMS GW Application Software (per MMS/s)	20 000	5	100 000
	Total, Software Licenses		**462 950**
Special Discounts			
10% Goodwill Discount			-46 295
ATI Interoperator GW Buyback Discount			-200 000
Closing Discount			-9 655
	Final Price, Software Licenses		**207 000**
Services			
Installation, integration & commissioning			60 000
Documentation			*Included*
Training			*Included*
	Final Price, Services		**60 000**
Software Support			
Support fee, annually on all installed Software Licenses			*8%*
Delivery Terms			
10 weeks after reception of Purchase Order			

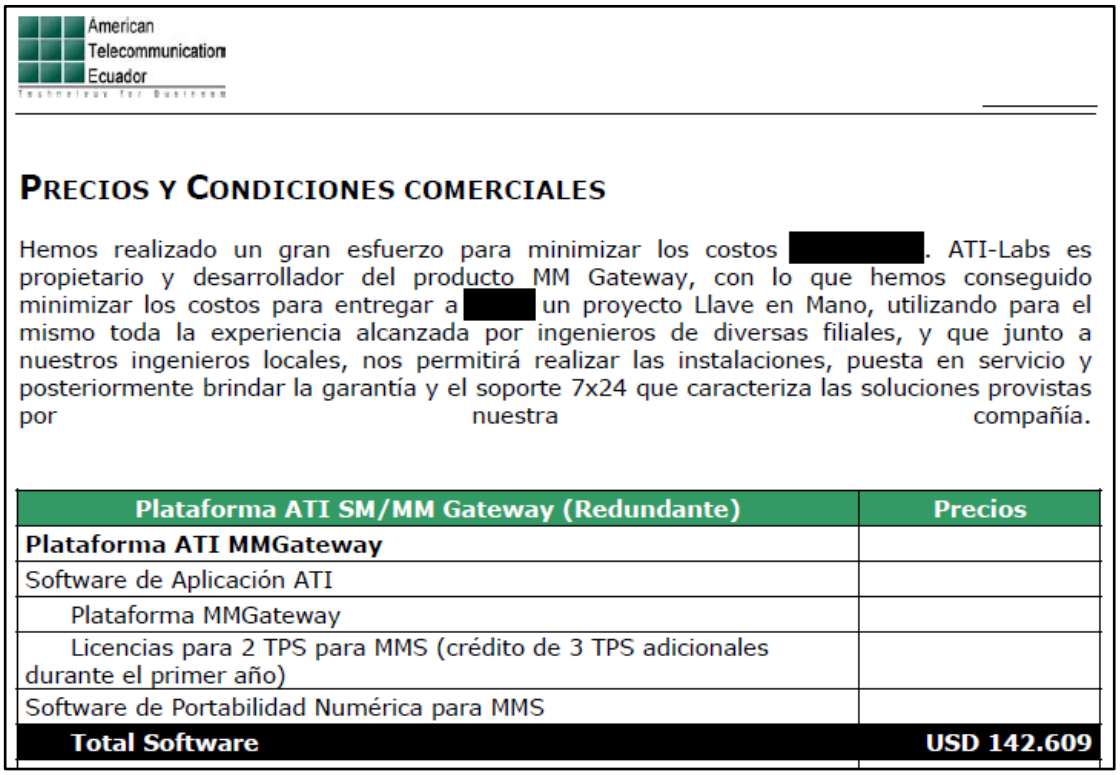
Fuente: Propuesta comercial digital enviada por Symsoft

Figura 5.8. Propuesta por el fabricante Symsoft

Se puede observar claramente un precio de USD 462.950,00 para el uso de software licenciado, tomando en cuenta que entre los componentes se encuentra las aplicaciones de SS7.

Propuesta ATI

Por otro lado tenemos a American Telecommunications, que compite con la primera para ganar con una propuesta muy parecida en características técnicas, Figura 5.9.



American Telecommunications Ecuador

PRECIOS Y CONDICIONES COMERCIALES

Hemos realizado un gran esfuerzo para minimizar los costos [redacted]. ATI-Labs es propietario y desarrollador del producto MM Gateway, con lo que hemos conseguido minimizar los costos para entregar a [redacted] un proyecto Llave en Mano, utilizando para el mismo toda la experiencia alcanzada por ingenieros de diversas filiales, y que junto a nuestros ingenieros locales, nos permitirá realizar las instalaciones, puesta en servicio y posteriormente brindar la garantía y el soporte 7x24 que caracteriza las soluciones provistas por nuestra compañía.

Plataforma ATI SM/MM Gateway (Redundante)	Precios
Plataforma ATI MMGateway	
Software de Aplicación ATI	
Plataforma MMGateway	
Licencias para 2 TPS para MMS (crédito de 3 TPS adicionales durante el primer año)	
Software de Portabilidad Numérica para MMS	
Total Software	**USD 142.609**

Fuente: Propuesta comercial digital enviada por ATI

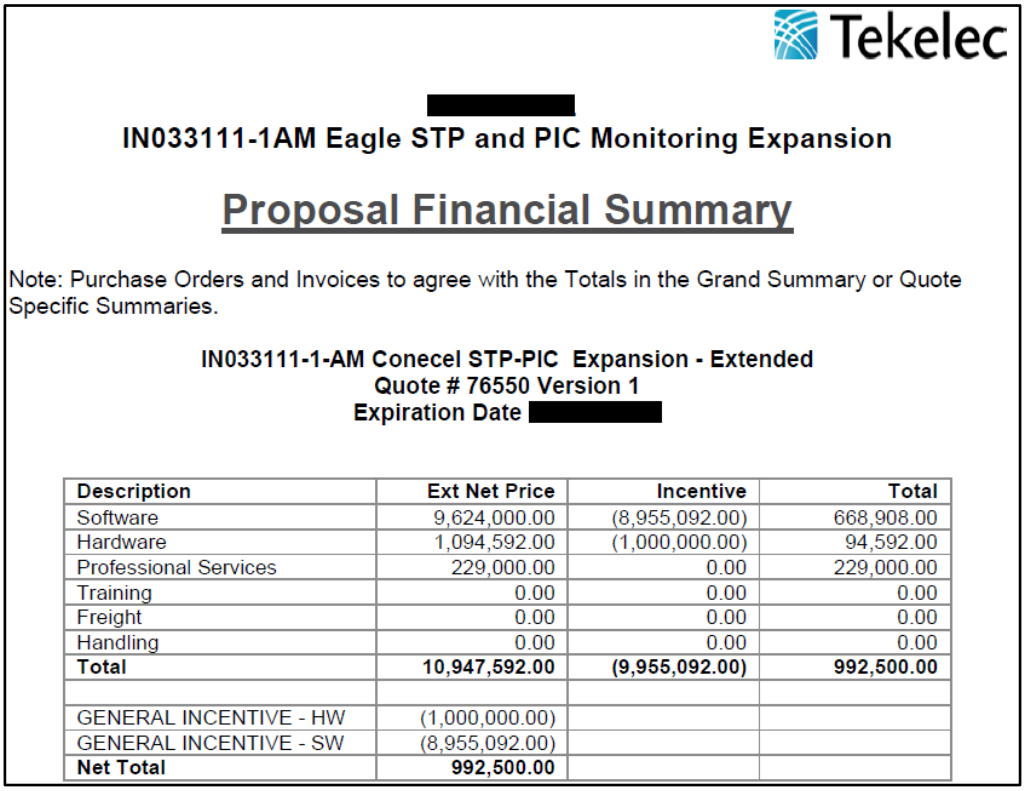
Figura 5.9. Propuesta comercial del participante ATI

Como se puede apreciar el monto para el software es de USD 142.609,00, precio muy por debajo de la propuesta anterior, que sin embargo se llegan a

equiparar a nivel de precio global con los ítems de hardware, servicios profesionales y capacitación.

Escenario 2: plataforma *STP Tekelec*, concentrador se señalización capaz de operar con todos los protocolos de señalización que son usados en el modelo SS7.

En este caso se trata de una expansión a realizarse del *Punto de Transferencia de Señal* (Figura 5.10).



Tekelec

[redacted]

IN033111-1AM Eagle STP and PIC Monitoring Expansion

Proposal Financial Summary

Note: Purchase Orders and Invoices to agree with the Totals in the Grand Summary or Quote Specific Summaries.

IN033111-1-AM Conecel STP-PIC Expansion - Extended
Quote # 76550 Version 1
Expiration Date [redacted]

Description	Ext Net Price	Incentive	Total
Software	9,624,000.00	(8,955,092.00)	668,908.00
Hardware	1,094,592.00	(1,000,000.00)	94,592.00
Professional Services	229,000.00	0.00	229,000.00
Training	0.00	0.00	0.00
Freight	0.00	0.00	0.00
Handling	0.00	0.00	0.00
Total	**10,947,592.00**	**(9,955,092.00)**	**992,500.00**
GENERAL INCENTIVE - HW	(1,000,000.00)		
GENERAL INCENTIVE - SW	(8,955,092.00)		
Net Total	**992,500.00**		

Fuente: Propuesta comercial digital enviada por Tekelec

Figura 5.10. Cotización de expansión PTS Tekelec

Para la expansión del STP se tiene que, luego del descuento a manera de incentivo que realiza, Tekelec, se tiene para el software un precio de USD 668.908,00.

Como se puede notar para estos 2 casos, las soluciones en donde viene ya integrada tanto la plataforma de señalización SS7 como la licencia para poder operar con los diferentes protocolos, supera fácilmente los 100 mil dólares americanos, variando entre uno y otro fabricantes así como también entre una y otra aplicación.

Mientras que, para la operación de una solución de *Código Abierto* como RHEL, se requiere el tener en primera instancia:

Subscripción a Red Hat

La suscripción a Red Hat ayuda a sus clientes aprovechar la posición que tiene Red Hat en la comunidad de desarrollo del *Código Abierto*. Esto permite a los clientes implantar sus infraestructuras tecnológicas más manejable y de menor coste que se les demande.

Mediante una subscripción se puede contar con:

- Novedades a nivel de software empresarial, incluidas opciones de seguridad, correcciones de errores, así como mejoras para todas las versiones soportadas.
- Soporte de hasta 10 años y mantenimiento para cualquier versión.

- Injerencia de los clientes en la innovación de la tecnología y la industria, gracias a las contribuciones exclusivas de Red Hat a la comunidad del *Código Abierto.*
- Derecho de acceso al programa *Red Hat Open Source Assurance*, que proporciona cobertura legal a los clientes que llevan a cabo desarrollos e implementaciones de *Código Abierto* [37]

Contrato de Soporte con Red Hat, Inc

Entre los beneficios cubiertos por este soporte se encuentra:

- Reporte de incidentes ilimitados.
- Disponibilidad 24x7.
- La retroalimentación en la resolución de incidentes está enfocada a la transferencia de conocimientos.
- Colaboración de varios proveedores para la identificación del problema de manera rápida.
- Guía y recursos en línea para la planificación, implementación y operación de la infraestructura de los clientes [38]

Estas dos figuras en la relación Cliente-Proveedor son plasmadas a través de un contrato económico que depende del número de servidores adquiridos, el tipo de contrato, versiones de las aplicaciones en el sistema operativo, entre otros. A continuación se muestra un par de ejemplos donde se reflejan los valores de los contratos con Red Hat.

ITEM	ESPECIFICACIONES SOLICITADAS POR LA GERENCIA	CANT.	VR UNITARIO PROMEDIO CON IVA	Vr. PROMEDIO CON IVA
	SOPORTE SISTEMAS OPERATIVOS LINUX RED HAT PARA FONADE			
1	• Suscripción RedHat Enterprise Linux Premium, para 9 servidores cada uno con un socket. El cual incluye: - Asistencia integral vía web y telefónica - Cobertura de 24 horas, 7 días a la semana - Tiempo de respuesta de 1 hora para incidencias críticas (4 horas para incidencias normales) - Sin límite en el número de incidencias - Actualización vía Red Hat Network	9	2836627	25529639
2	Contrato de soporte en sitio, anual, mínimo 200 horas.	200	101674	20334700
	TOTAL COTIZACION			**45.864.339**

Figura 5.11 Contrato de Soporte entre Red Hat, Inc y el Fondo Financiero de Proyectos de Desarrollo de Colombia

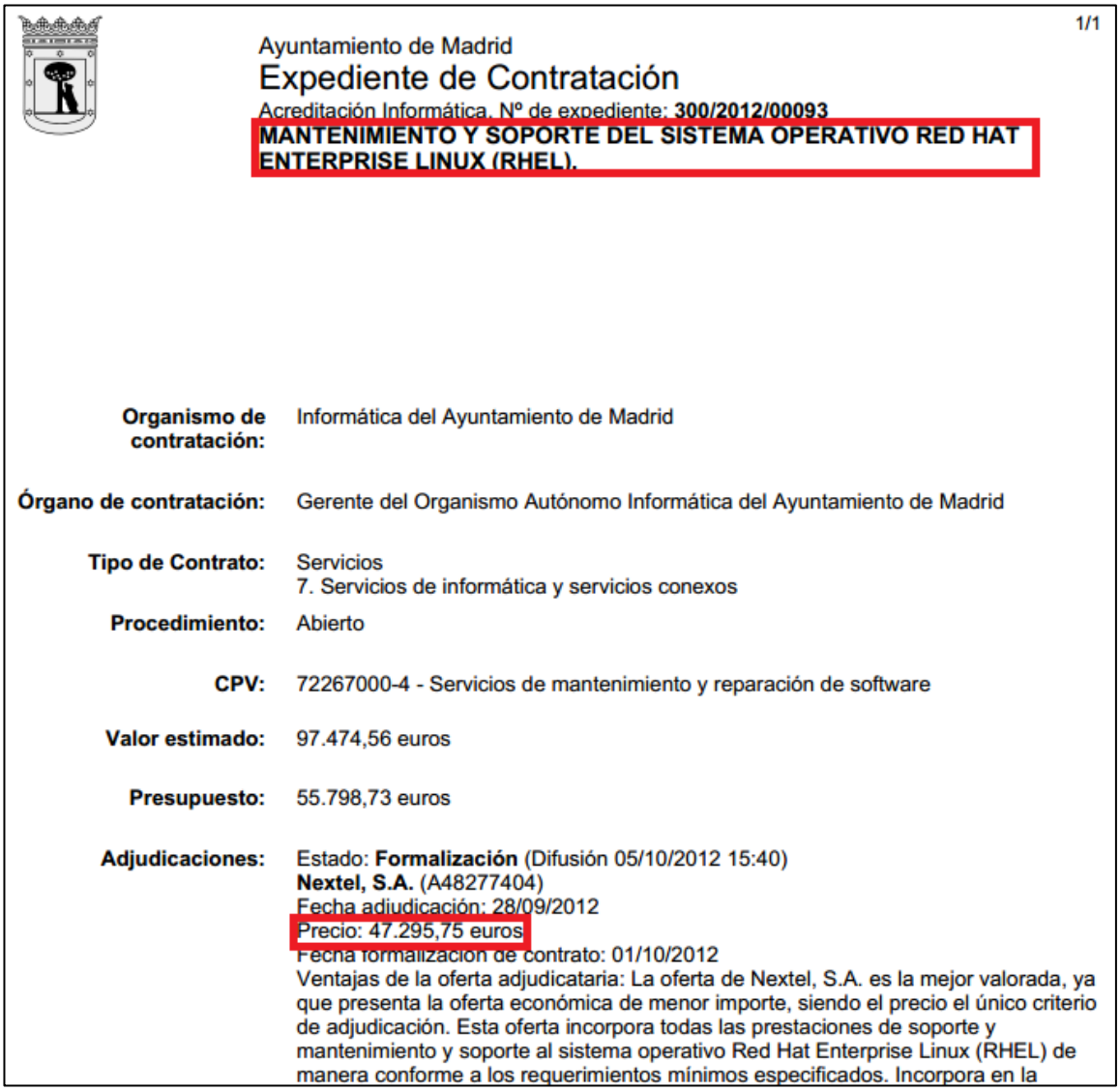
En la Figura 5.11 se puede valorar que el monto acordado entre el Fondo Financiero de Proyectos de Desarrollo de Colombia (FONADE) y Red Hat, Inc, por el soporte *Premium* para 9 servidores con RHEL es de 45’864.339 pesos colombianos (alrededor de USD 25.480,00) [39]

Donde el soporte *Premium* de Red Hat es especificado por la Figura 5.12.

		Soporte Premium	***Soporte Standard***
Severidad 1	Tiempo de respuesta	1 hora de oficina	1 hora de oficina
	Tiempo de resolución	8 horas de oficina	2 días laborables
Severidad 2	Tiempo de respuesta	2 horas de oficina	4 horas de oficina
	Tiempo de resolución	2 días laborables	2 días laborables
Severidad 3	Tiempo de respuesta	4 horas de oficina	1 día laborable
	Tiempo de resolución	2 días laborables	4 días laborables
Severidad 4	Tiempo de respuesta	8 horas de oficina	2 días laborables
	Tiempo de resolución	4 días laborables	4 días laborables

Figura 5.13 Tipo de soportes de Red Hat, Inc [40]

Por otro lado, en la Figura 5.13 se tiene la publicación del contrato acordado entre Red Hat y el Ayuntamiento de Madrid por el soporte de sus servidores:



1/1

Ayuntamiento de Madrid

Expediente de Contratación

Acreditación Informática. Nº de expediente: **300/2012/00093**

MANTENIMIENTO Y SOPORTE DEL SISTEMA OPERATIVO RED HAT ENTERPRISE LINUX (RHEL).

Organismo de contratación:	Informática del Ayuntamiento de Madrid
Órgano de contratación:	Gerente del Organismo Autónomo Informática del Ayuntamiento de Madrid
Tipo de Contrato:	Servicios 7. Servicios de informática y servicios conexos
Procedimiento:	Abierto
CPV:	72267000-4 - Servicios de mantenimiento y reparación de software
Valor estimado:	97.474,56 euros
Presupuesto:	55.798,73 euros
Adjudicaciones:	Estado: **Formalización** (Difusión 05/10/2012 15:40) **Nextel, S.A.** (A48277404) Fecha adjudicación: 28/09/2012 Precio: 47.295,75 euros Fecha formalización de contrato: 01/10/2012 Ventajas de la oferta adjudicataria: La oferta de Nextel, S.A. es la mejor valorada, ya que presenta la oferta económica de menor importe, siendo el precio el único criterio de adjudicación. Esta oferta incorpora todas las prestaciones de soporte y mantenimiento y soporte al sistema operativo Red Hat Enterprise Linux (RHEL) de manera conforme a los requerimientos mínimos especificados. Incorpora en la

Figura 5.13 Contrato de Soporte entre Red Hat, Inc y el Ayuntamiento de Madrid [41]

El contrato aceptado es por el mantenimiento y soporte del sistema operativo Red Hat Enterprise Linux utilizado en diferentes servidores instalados en los centros de proceso de datos de Informática del Ayuntamiento de Madrid y del Área de Gobierno de Seguridad (Dirección General de Emergencias y

Protección Civil-Samur y Bomberos) sobre los que se soportan, por un valor de 47.295,75 euros (alrededor de USD 61.484,475) [41]

Con lo expuesto anteriormente, se puede notar lo difícil que es realizar un análisis comparativo entre soluciones iguales propietarias contra las de código abierto, sin embargo, pese a esta dificultad, es posible concebir una idea general de lo convenientes (a nivel costos) que son las alternativas de *Código Abierto.*

No obstante, con la idea de realizar una comparación de escenarios similares la empresa *Bearing Point* realiza un análisis comparativo no de soluciones sino más bien entre el costo de Licencias de servidores propietarios versus el costo del soporte/subscripción de RHEL para la misma cantidad de equipos y su extracto se muestra a continuación en las Figuras 5.15 y la Figura 5.16 [42]

Detalles de precios de Microsoft						
	Empresas medianas			Empresas grandes		
	Cant.	Publicado	Cotizado	Cant.	Publicado	Cotizado
Año 1						
Licencias de la Edición Estándar	26	25,974	5,261	470	469,530	81,310
Licencias de la Edición Enterprise	3	11,997	2,042	52	207,948	32,951
Licencias de acceso de cliente adicionale	232	9,280	1,933	5742	229,680	40,194
Software Assurance	N/A	-	6,927	N/A	-	115,841
Soporte Essential	N/A	8,000	8,000	N/A	-	-
Soporte Premier	N/A	-	-	N/A	50,000	50,000
Totales Año 1		$ 55,251	$ 24,163		$ 957,158	$ 320,296

Figura 5.15 Costo por Windows Server 2003.

Detalles de precios de Red Hat						
	Empresas medianas			Empresas grandes		
	Cant.	Publicado	Obtenido	Cant.	Publicado	Obtenido
Año 1						
Suscripciones de ES Standard	26	$ 20,774	$ 15,574	470	$ 375,530	$ 281,530
Suscripciones de AS Premium	3	7,497	5,250	52	129,948	91,000
Módulo de adm. de Red Hat Network	29	2,233	2,233	522	40,194	40,194
Nodos de agrupación de Red Hat	0	-	-	4	1,996	1,996
Totales Año 1		$ 30,504	$ 23,057		$ 547,668	$ 414,720

Figura 5.16 Costo por Red Hat Enterprise Linux 3.

Adicionalmente, a continuación se adjunta un cuadro comparativo de las propuestas comerciales para los servicios de MMS GW mostradas anteriormente, versus el costo del mismo aplicativo desarrollado sobre sistemas abiertos como es el Red Hat Enterprise Linux. La Figura 5.17 logra mostrar lo conveniente que resulta el elegir desarrollos sobre Sistemas Abiertos, en este cuadro se nota que la opción de implementar el aplicativo MMS GW sobre 9 servidores RHEL es 7 veces menor contra la propuesta más económica de un sistema propietario.

	Sistemas propietarios		Sistemas Abiertos
	Symsoft	ATI	9 servidores Red Hat Enterprise Linux
Licencia de SW de MMS GW	$ 112.950,00	$ 142.609,00	$ -
Licencia por tráfico MMS	$ 100.000,00		$ -
Subscripción RHEL			$ 14.183,13
Mantenimiento Anual	$ 66.988,00	$ 36.560,00	$ 11.297,06
Total	$ 279.938,00	$ 179.169,00	$ 25.480,19

Figura 5.17 Cuadro comparativo para la solución MMS-GW.

Por lo plasmado en este apartado, es fácil inferir que, plataformas, soluciones, productos y/o servicios desarrollados sobre arquitecturas *Código Abierto* representan una alternativa muy válida a la hora de decidir ante una propietaria. Desde luego que el análisis para tomar decisiones donde la relación costo/beneficio debe ser muy alta, no solo debe basarse en el costo del producto, sino que también, debe ser la consecuencia de la profunda valoración de las cualidades y bondades técnicas, ambientales y económicas.

5.4 Discusión

De lo que se puede apreciar en el apartado anterior, existen muchos aspectos a analizar dentro de las pruebas que se realizan para cada uno de los paquetes instalados.

En primera instancia se tiene que, para que las pruebas sean aplicables, es necesario establecer al menos una asociación SCTP, es decir que a nivel del protocolo de transporte SCTP exista una conexión con otra entidad de señalización como lo podría ser un signaling gateway, con el que se pueda establecer una asociación SCTP y pueda fluir el intercambio de información de este protocolo a través de sus *chunk's* (unidades de señalización a nivel de SCTP).

Otro punto a considerar está relacionado con el nivel de conocimiento acerca de los aspectos de SCTP, las pruebas ejecutadas concernían a los

controladores de las interfaces NPI y NTI, de tal manera que el conocer el modo de operar y funciones relacionados permitan realizar actividades para solucionar los problemas encontrados, para ello la revisión de los estándares y recomendaciones es imperativo.

Por otro lado, los errores encontrados antes de la ejecución de las pruebas giran alrededor de temas particulares del manejo de Linux, entre los más comunes se tiene:

- Falta de permisos de ejecución de los scripts, lo que significa que al momento de ejecutar el script se generaba un error haciendo alusión a no tener los suficientes privilegios para ejecutar tal script, por ello durante las pruebas se optó por operar con el súper usuario de Linux, root.
- Archivos no encontrados necesarios para la ejecución del script, el error generado es que, durante la ejecución de la prueba, el script requería de información contenida en un determinado archivo y el script al buscarlo no lo encontraba en el directorio señalado dentro de la programación del script, esto se superó enviando a buscar el archivo referenciado y luego copiándolo en el directorio adecuado, otra manera de solucionar este problema, fue el editar el script, corrigiendo el directorio donde si era posible alcanzar el archivo buscado.

Así mismo, se debe comentar que este comportamiento y resultado de las pruebas fue logrado tanto en el servidor OpenSS7 implementado como en la máquina virtual.

CONCLUSIONES

Del presente trabajo se puede inferir aspectos de mucho interés a los involucrados en la implementación y/o desarrollo de sistemas abiertos, en particular en el mundo de la telefonía.

1. El SS7 está posicionado en las operadoras telefónicas como principal sistema usado de manera amplia en sus redes, ya que, gracias a este sistema se han podido desarrollar servicios y productos que representen ingresos significativos para las dichas operadoras. Los servicios que se pueden implementar sobre SS7, más destacados en este trabajo, son los SMC, IN o servicio de Prepago y consultas a bases de datos.
2. El sistema operativo Linux, desarrollado por el proyecto GNU, es el sistema usado por el proyecto OpenSS7, para implementar sobre esta plataforma todos los diferentes niveles o capas de SS7, SIGTRAN y VoIP,

en particular las distribuciones de Linux con versión del kernel 2.4 y 2.6, esto debido a la particularidad de que en estas versiones se encuentra pre-incorporada las facilidades *stream*s de SCTP (pilar fundamental para establecer una asociación SCTP). Adicionalmente, el hecho de que Linux sea un sistema operativo de *Código Abierto*, permite desarrollar módulos de compatibilidad para que pueda soportar nuevos componentes, en este caso los de OpenSS7.

3. Los niveles de SS7 instalados en el servidor *stand-alone* e integrados al kernel 2.6 de la distribución de Linux Fedora Core 9 a manera de componentes de software, permitieron erigir una plataforma estable con la que se logra la interacción entre niveles de los diferentes protocolos de SS7, SIGTRAN y VoIP, aun cuando en la implementación no se llegó a probar la interoperabilidad de estas capas instaladas con las de otro servidor de características similares a través de una red local.
4. La correcta configuración y operación de OpenSS7 permite desarrollar aplicaciones *in-house* que, obtenidas de manera comercial, resultan mucho más impactantes a las cuentas de CAPEX y OPEX de las operadoras de telefonía, advirtiendo también, que existen rubros de mantenimientos que deben ser cubiertos a manera de contratos por más libre que sea el sistema adquirido, este es el caso del sistema operativo Linux en la distribución de Red Hat, no se cobra la licencia del sistema operativo como tal, pero si el soporte de mantenimiento.

RECOMENDACIONES

Las recomendaciones a continuación citadas, van encaminadas a lograr una correcta instalación de los paquetes así como también de comprender el alcance de cada componente instalado:

1. Instalar el sistema operativo anfitrión de los paquetes OpenSS7 de acuerdo a los requisitos demandados por el proyecto OpenSS7, en cuanto a arquitectura, versión y herramientas de programación [5]
2. Se recomienda tener un nivel avanzado de Linux para poder manipular, instalar, e incluso llegar a editar los programas ejecutables de los componentes. Así mismo, para los archivos no pre-configurados bajo el directorio /etc/ alusivos al proyecto OpenSS7, se requiere de un alto conocimiento de programación para lograr completar el formato y sintaxis

de acuerdo a lo que demandan los archivos de programación *.c* y los de cabecera *.h*.

3. Para la compilación de los paquetes a instalar, es necesario tomar en cuenta que el compilador GCC sea la misma versión con la que fue construido el Kernel del sistema operativo.
4. El lograr una adecuada implementación de los componentes OpenSS7 depende de los conocimientos que se dispongan acerca de los protocolos de señalización, como SCTP, M3UA, ISUP… por lo que se aconseja estudiar profundamente las recomendaciones y estandarizaciones de la ITU y ANSI acerca de estos protocolos como de los diferentes aspectos relacionados.
5. Con el afán de poder probar un correcto e íntegro desempeño de los componentes instalados, se recomienda implementar un segundo servidor y conectarlos a través de una *Red de Área Local (Local Area Network, LAN)* para, bajo esta topología, ejecutar los set de pruebas en cada nivel de SS7.
6. El desarrollo de sistemas abiertos es una iniciativa que ofrece un panorama muy amplio para implementaciones tecnológicas, en el campo de las telecomunicaciones se pueden desarrollar plataformas de servicio de valor agregado como:
    - Aplicativos SMPP para el envío/recepción de SMS's en lenguajes tradicionales como *Hypertext Preprocessor* (PHP) y Visual Basic.

- Desarrollo de plataforma SMSC con modalidad Store & Forward, la cual puede ser probada con muchas de las herramientas para probar el protocolo SMPP encontradas en la WEB.
- Servicio denominado "*Llámame*", el cual consiste en detectar, a través de señalización SS7, cuando un abonado A ha intentado llamar a un abonado B, a este último le llegara un mensaje de texto diciendo que el número A desea que le devuelva la llamada.
- Plataforma *Open Wireless Application Protocol* (OWAP), que es el protocolo de aplicaciones inalámbricas para teléfono móviles a servicios de internet

BIBLIOGRAFÍA

[1] Open Source Initiative, The Open Source Definition, http://opensource.org/osd, fecha de consulta Diciembre 2012

[2] Proyecto GNU, Visión general del sistema GNU, http://www.gnu.org/gnu/gnu-history.es.html, fecha de consulta Diciembre 2012

[3] Proyecto GNU, GNU General Public License, http://www.gnu.org/copyleft/gpl.html, fecha de consulta Diciembre 2012

[4]OpenSS7 Corporation, Overview, http://www.openss7.org/overview.html, fecha de consulta Diciembre 2012

[5] OpenSS7 Corporation, System Requirements, http://www.openss7.org/sysreq.html, fecha de consulta Diciembre 2012

[6] Levine E., Abdellah F.G., 1994, “Preparing Nursing Research for the 21st Century. Evolution. Methodologies, Chalges”. New York, USA.

[7] Russel Travis, 2006. “Signaling System #7”. New York, USA.

[8] Van Bosse & Devetak, 2007, pag 167, “Signaling in Telecommunication Networks”, New Yersey, USA.

[9] Davidson Jonathan, 2006, pag 96, “Voice over IP Fundamentals”, Indianapolis, USA.

[10] Javvin Technologies, 2004, pag. 314, “Network Protocols handbook”, Saratoga, USA

[11] Javvin, SIGTRAN, https://javvin.com/protocolSIGTRAN.html, fecha de consulta Diciembre 2012

[12] IEFT, Network Working Group Request for Comments: 3094, http://tools.ietf.org/rfc/rfc3094.txt, fecha de consulta Enero 2013

[13] IEFT, Network Working Group Request for Comments: 3057, http://www.ietf.org/rfc/rfc3057.txt, fecha de consulta Enero 2013

[14] OpenSS7 Corporation, rfc4165, http://www.openss7.org/rfc/rfc4165.txt, fecha de consulta Enero 2013

[15] OpenSS7 Corporation, rfc3331, http://www.openss7.org/rfc/rfc3331.txt, fecha de consulta Enero 2013

[16] IEFT, Network Working Group Request for Comments: 3332, http://tools.ietf.org/html/rfc3332, fecha de consulta Enero 2013

[17] OpenSS7 Corporation, draft-hou-sigtran-tua-00, http://www.openss7.org/internet-drafts/draft-hou-sigtran-tua-00.txt, fecha de consulta Enero 2013

[18] IEFT, Network Working Group Request for Comments: 4960, http://www.ietf.org/rfc/rfc4960.txt, fecha de consulta Enero 2013

[19] Laurent Andrew, 2004, pag 10-11, “Understanding Open Source and Free Software Licensing”, Sebastopol, USA.

[20] Alegre Maria del Pilar, 2012, pag 84-99, “Sistemas Operativos Monopuestos”, Madrid, España.

[21] RADCOM, Fine - Tuning Voice over Packet Services, http://www.protocols.com/papers/voip2.htm, fecha de consulta Diciembre 2012

[22] www.dspace.ups.edu.ec/bitstream/123456789/210/3/Capitulo%202.pdf

[23] Tracey Marion, 2008, Pag 40, “VOiP Voice Over IP 100 Success Secrets”, Londres, Inglaterra.

[24] Wallingford Theodore, 2005, Pag 34, “Switching to VoIP”, Sebastopol, USA.

[25] OpenSS7 Corporation, Mission, http://www.openss7.org/mission.html, fecha de consulta Enero 2013.

[26] OpenSS7 Corporation, Programs, http://www.openss7.org/programs.html, fecha de consulta Enero 2013.

[27] OpenSS7 Corporation, Linux Fast-STREAMS, http://www.openss7.org/streams_man.html, fecha de consulta Enero 2013.

[28] Stewart & Conrad, SCTP: AnOverview, http://www.sctp.org/SCTPOttawaPart1v1.2.pdf, fecha de consulta Enero 2013.

[29] La Monte H.P., History, http://lksctp.sourceforge.net/history.html, fecha de consulta Enero 2013.

[30] Krishnamurthy, Seagren, Alder, Bayles, Burke, Carter, Faskha, 2008, Pag 18, “How to cheat at Securing Linux”, Burlington, USA.

[31] OpenSS7 Corporation, Hardware Selection Guide, http://www.openss7.org/device_hw.html, fecha de consulta Enero 2013.

[32] Howtoforge, The Perfect Server - Fedora 9, http://www.howtoforge.com/perfect-server-fedora9.

[33] RosBusinesConsulting, Mirrors, http://fedora-mirror01.rbc.ru/pub/fedora-archive/fedora/linux/releases/9/Fedora/i386/iso/, fecha de consulta Diciembre 2012.

[34] OpenSS7 Corporation, PublicKeys, http://www.openss7.org/pubkeys.html, fecha de consulta Diciembre 2012.

[35] González Víctor, El compilador GCC, http://iie.fing.edu.uy/~vagonbar/gcc-make/gcc.htm, fecha de consulta Enero 2013.

[36] OpenSS7 Corporation, Operating System, http://www.openss7.org/os_man.html, fecha de consulta Enero 2013.

[37] Red Hat, Inc, Benefits of a Red Hat subscription, http://www.redhat.com/support/subscription-benefits/, consultado en Marzo 2013.

[38] Red Hat, Inc, Red Hat Partner Support, http://www.redhat.com/support/red-hat-support-partners.html, consultado en Marzo 2013.

[39] FONADE, Contrato de Soporte, http://www.fonade.gov.co/Contratos/Documentos/2343__20101209054218A

NEXO%2001-%20INFORMACION%20GENERAL%20MC10-441-2010.pdf, consultado en Marzo 2013.

[40] Ayuntamiento de Madrid, PLIEGO DE PRESCRIPCIONES TÉCNICAS PARTICULARES, http://dr-www.munimadrid.es/UnidadesDescentralizadas/PerfilContratante/PC_OrganismosAutonomos/PC_IAM/PROCEDIMIENTOS%20ABIERTOS/2012/ficheros/PPTP%20300-2012-00093.pdf, consultado en Marzo 2013.

[41] Ayuntamiento de Madrid, Perfil del Contratante, http://dr-www.munimadrid.es/portales/munimadrid/es/Inicio/Ayuntamiento/Hacienda/Perfil-de-Contratante/Expedientes-Adjudicados/Mantenimiento-y-soporte-del-sistema-operativo-Red-Hat-Enterprise-Linux-(RHEL).?vgnextfmt=default&vgnextoid=602242ba04078310VgnVCM1000000b205a0aRCRD&vgnextchannel=f939ec5228488110VgnVCM100000c90da8c0RCRD, consultado en Marzo 2013.

[42] Bearing Point, Comparación de Costos de Licenciamiento y Soporte, http://www.microsoft.com/conosur/hechos/analyses/comparable.mspx, consultado en Marzo 2013.

ANEXO A

Mensajes de Señalización

A continuación se realiza un breve resumen de los mensajes de señalización y protocolos empleados en la realización de una llamada de una línea celular prepago.

Los servicios para los abonados prepago son facturados en línea por la plataforma de *Red Inteligente*, la misma que usa los protocolos de aplicación: CAMEL para cobrar llamadas de voz, y DIAMETER/CAP3 para facturar servicios de datos como SMS, GPRS, Blackberry, Internet… El ejemplo que se describe a continuación es exclusivamente de un abonado que al presionar SEND en su equipo móvil, la solicitud de la llamada llega a la *Central de Conmutación Móvil (Mobile Switching Center, MSC*) y esta envía la consulta a la plataforma de *Red Inteligente* por la autorización de conectar la llamada en el caso de contar con el saldo necesario (Figura A.1).

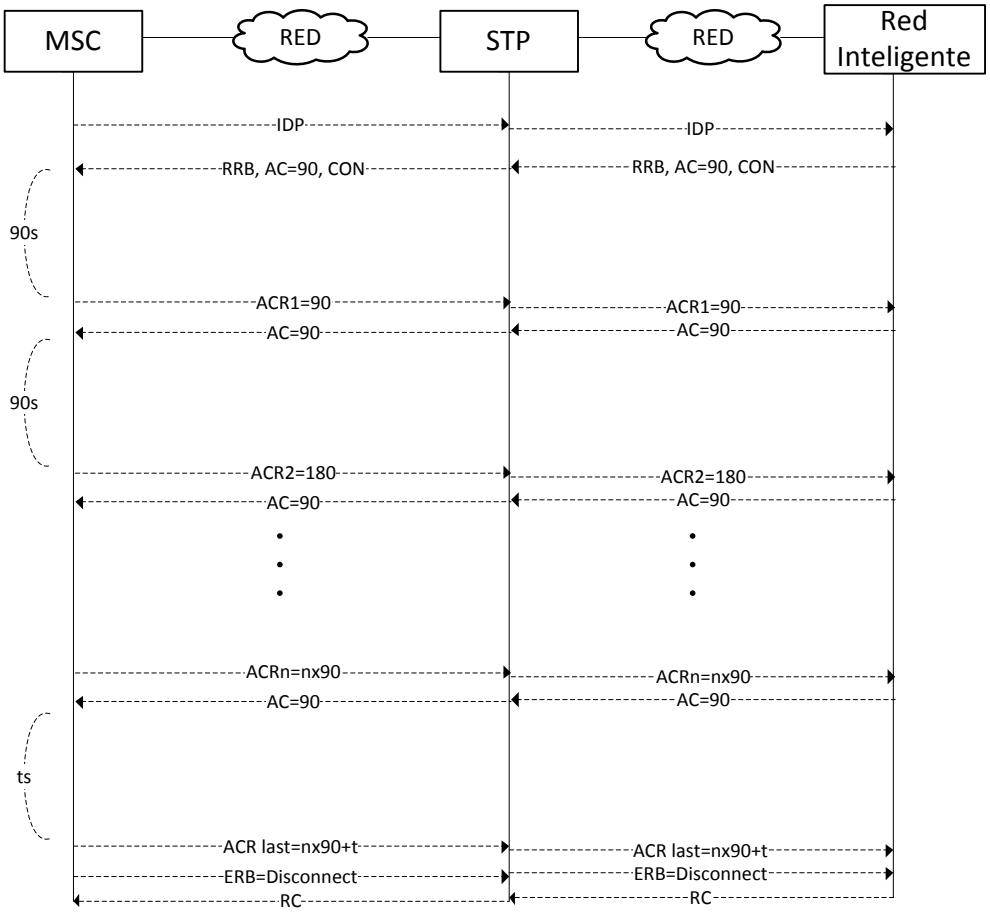


Figura A.1. Flujo de mensajes CAMEL

Cuando el abonado presiona la tecla para ejecutar la llamada, el requerimiento llega a la MSC (para las llamadas GSM), y ésta identifica que el intento de llamada proviene de un abonado prepago, por lo que envía la consulta a la plataforma Prepago vía señalización CAMEL, para obtener la autorización de conectar la llamada en caso de tener saldo suficiente, así:

1. El primer mensaje CAMEL que envía la MSC, es el *Initial Detection Point* (*IDP*), el cual lleva información acerca del número del abonado que

intenta realizar la llamada (#A) así como el número de la línea a la cual desea llamar (#B).

2. Los mensajes que se intercambien entre la MSC y la *Red Inteligente*, pasan a través del PTS, el cual es concentrador de señalización de la red celular.
3. La información que arriba a Prepago en el IDP, sirve para poder identificar la tarifa a cobrar por la llamada (con el #B) y la cuenta desde donde se debita aquella tarifa (#A).
4. En el caso de que el #A cuente con el saldo suficiente para realizar aquella llamada, Prepago, responde con 3 mensajes CAMEL:
    - *Request Report BCSM Event*: con el cual la *Red Inteligente* le solicita a la MSC que monitoree la llamada, ante la posibilidad de otros eventos entrantes o salientes relacionados a esos abonados.
    - *Apply Charging:* este mensaje le indica el periodo por el cual la llamada puede estar enlazada sin que la MSC tenga la necesidad de volver a consultar si prosigue la llamada, para la llamada del ejemplo, la operadora ha configurado en 90s este periodo.
    - *Connect:* mensaje que le indica a la MSC que autoriza la conexión de la llamada.
5. Luego del procesamiento a nivel de SS7 (MAP) que realiza la MSC para conectar la llamada en la red GSM con el #B, la llamada es enlazada y la

conversación perdura por el tiempo especificado en el *Apply Charging*, 90s.

6. Una vez cumplido los 90s, la MSC envía el mensaje CAMEL:
    - *Apply Charging Report:* mensaje que le indica a la *Red Inteligente* que se cumplieron 90s.
7. Prepago responde con un nuevo *Apply Charging* de 90s (en caso de que el abonado #A aun tenga suficiente saldo para una llamada de 90s a ese destino).
8. Y el ciclo se repite hasta que se consuma el saldo que le queda al abonado #A o como se muestra en la Figura A.1, el abonado #A o #B cierran la llamada de manera voluntaria.
9. Cumplido lo anterior, la MSC envía el último *Apply Charging Report* con el tiempo total de la conversación, para que esa cantidad finalsea debitada por Prepago, junto al mensaje *ERB=Disconnect* con lo que le indica que no está esperando otro *Apply Charging*, ya que la llamada ha sido terminada.
10. La *Red Inteligente* responde con el último mensaje: *Release Call*, cerrando de manera formal la conversación CAMEL que fue iniciada con un IDP, los circuitos son liberados de lado de la MSC y la cuenta del abonado #A es mermada en la cantidad de tiempo que mantuvo la conversación con el abonado #B.

Mediante el flujo de mensajes CAMEL antes descrito, se obtiene un control de las llamadas que realizan los abonados prepago de una red celular, sin embargo el protocolo CAMEL es de aplicación (opera sobre TCAP en el modelo SIGTRAN), tal como se lo mostró en la Figura 2.5, en la que los protocolos que operan debajo son IP, SCTP, M3UA, SCCP y TCAP.

A continuación, se procede con una somera revisión de los mensajes que envía cada uno de estos protocolos al enviar un IDP, empezando el relato desde SCTP como nivel más bajo para el análisis.

A nivel SCTP, se puede observar en la traza tomada, que existe el paquete llamado *HEARBEAT* o lo que es conocido a nivel de SCTP: un *CHUNK HEARBEAT*, el mismo que es enviado entre dos puntos de señalización (desde la entidad con IP: 192.168.216.72 hacia 192.168.188.25) y se lo utiliza para determinar que el camino o asociación SCTP esta expedito, y podría ser usado para transportar información de protocolos de las capas superiores.

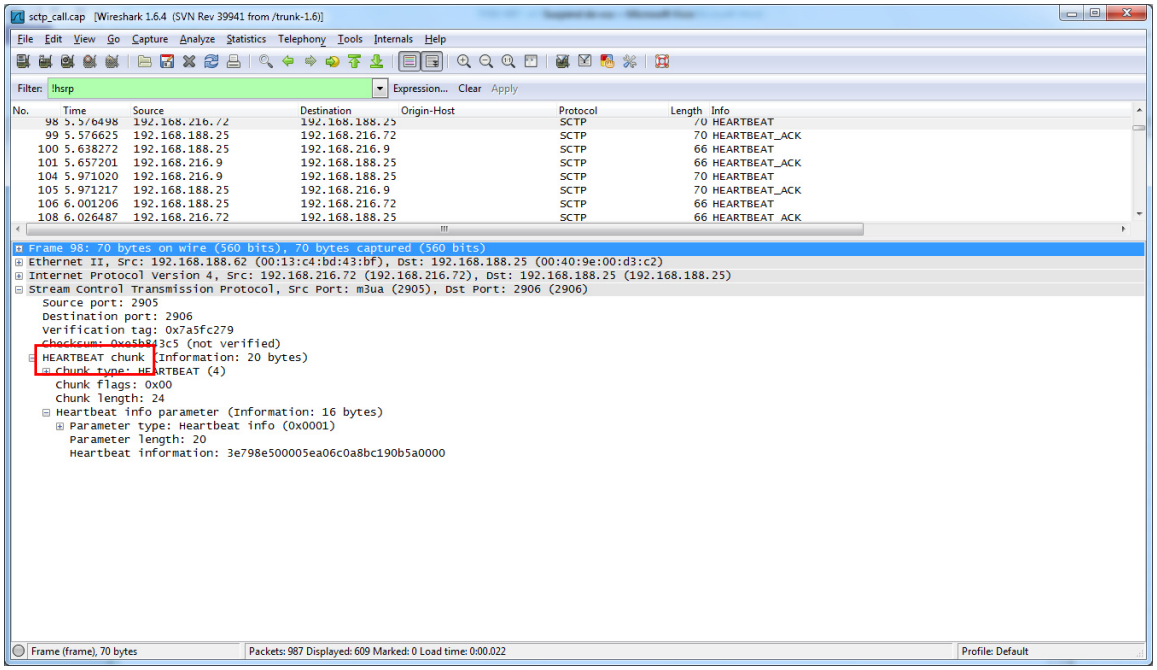


Figura A.2. Chunk HEARBEAT de SCTP

La respuesta al paquete 98, es el paquete 99, el mismo que se envía luego de alrededor de una diez milésima de segundo, la respuesta se la puede observar en la Figura A.3, que muestra que la entidad de señalización con IP: 192.168.188.25 (*Signaling Gateway* de la plataforma Prepago) responde satisfactoriamente con una HEARBEAT_ACK al paquete HEARBEAT previamente enviado por 192.168.216.72 (PTS de la red celular).

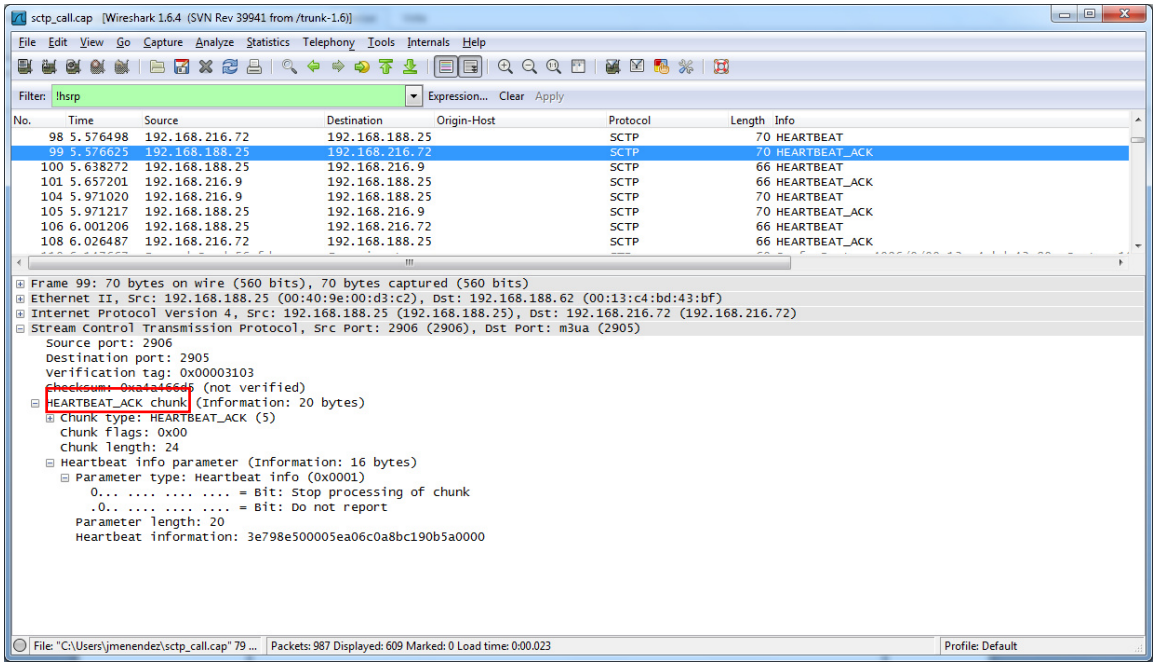


Figura A.3. Chunk HEARBEAT_ACK de SCTP

Los mensajes de SCTP revisados son intercambiados constantemente entre las partes involucradas, para asegurarse que no existan inconvenientes a la hora de transportar datos de protocolos de los niveles superiores.

Cuando existe un paquete de información que debe ser transportado hacia el destino (*SG* de Prepago en este caso), la unidad de señalización que usa SCTP para enviar dicha información es el *DATA CHUNK*, sin embargo esta no es la única información que es transportada, ya que existen más parámetros que le interesan a ambas entidades (Figura A.4), como lo son:

- *Source Port*: puerto SCTP de origen.
- *Destination Port*: puerto SCTP de destino.

- *Transmission Sequence Number* (*TSN*): número secuencial para la transmisión.
- *Stream Identifier*: identificador del *stream* dentro de la asociación SCTP.
- *Payload Protocol Identifier*: identifica que tipo de información (protocolo) esta llevando el *Data Chunk.*

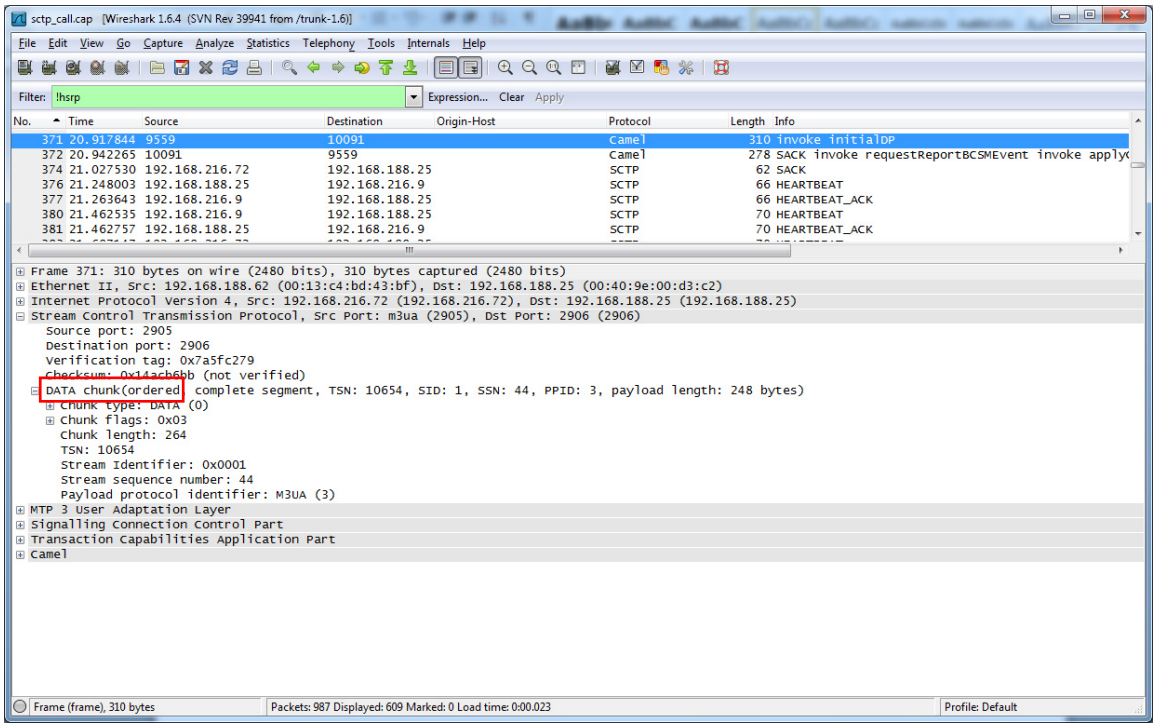


Figura A.4. Chunk DATA de SCTP que transporta un paquete M3UA

Como se puede observar de la figura anterior, el paquete que se está transportando es de M3UA (ver *Payload Protocol Identifier*), y como se lo muestra en la Figura A.5, este paquete lleva información afinada acerca del origen y destino:

- *OPC*: punto de código del origen.
- *DPC*: punto de código del destino.
- *Service Indicator* (*SI*): indicador del servicio que está transportando.
- *Network Indicator* (*NI*): indicador de la red.
- *Message Priority* (*MP*): prioridad de mensaje.
- *Signalling Link Selection* (*SLS*): indica cual es el enlace de señalización usado.

Con esta información, los niveles superiores pueden ir recuperando la información final, para el caso mostrado el siguiente protocolo a explorar es el SCCP, según lo que indica el campo SI=3 (SCCP).

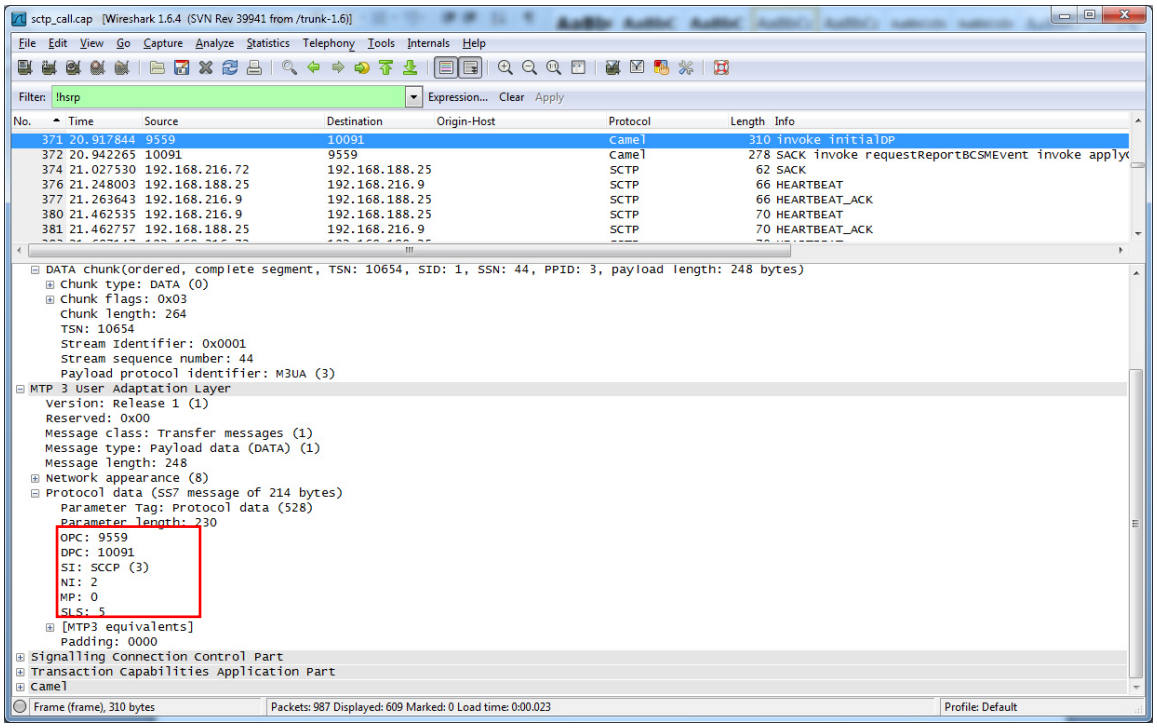


Figura A.5. Paquete M3UA

Como es sabido, SCCP es usado para las consultas directas a base de datos y el encaminamiento se lo realiza a nivel de *Global Title* (*GT*), motivo por el cual, en la Figura A.6, se pueden observar que los principales campos son:

- *Subsystem Number*: número que indica el subsistema usado en el nivel superior.
- *GT Called Party Address:* identifica al GT al que se va a realizar la consulta.
- *GT Calling Party Address:* identifica al GT desde donde se realiza la consulta.

Para el ejemplo, el GT que realiza la consulta a la base de datos de Prepago es el 59397999126, el cual pertenece a la MSC, identificada por el GT 59397999104, y la consulta es invocada a través del subsistema 146, es decir *CAMEL Application Part* (*CAP*).

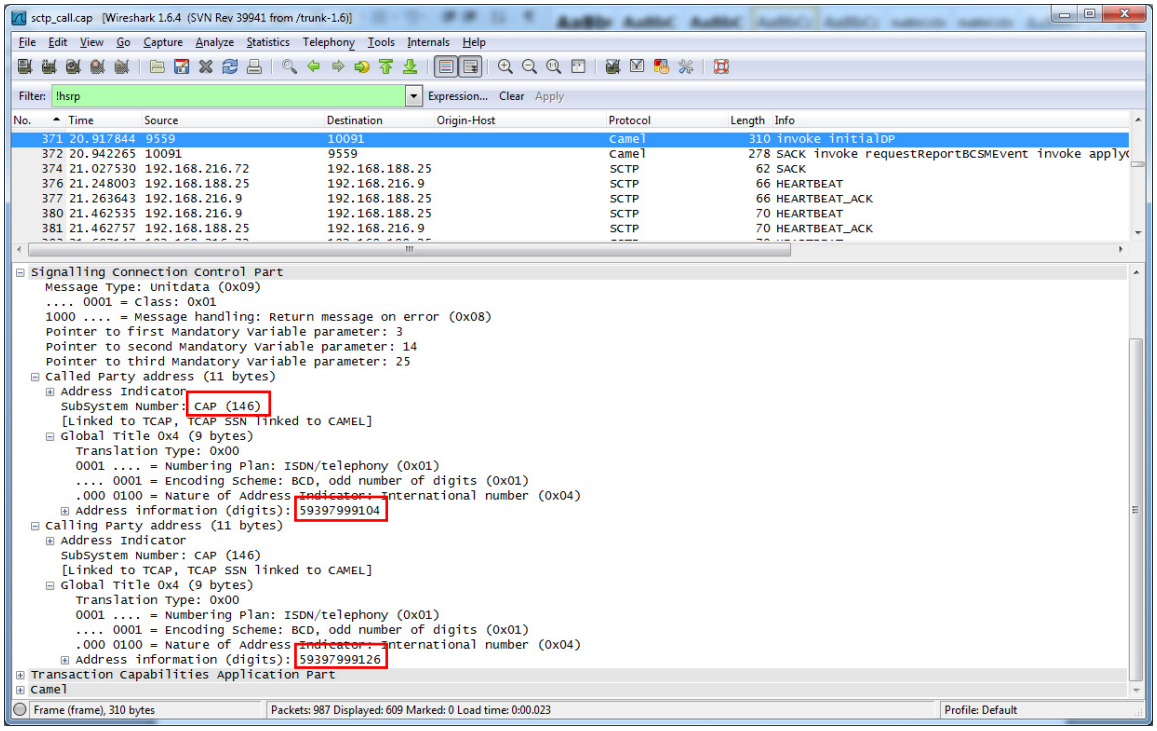


Figura A.6. Información enviada a nivel SCCP

Con los mensajes de los protocolos revisados, la información llega al extremo de interés, el destino, y está lista para ser leída por la entidad que recibe la consulta. Para este caso, la *Red Inteligente* lee la información CAMEL a través de TCAP, donde se encuentra el mensaje *begin* que indica el inicio de una conversación a nivel de CAMEL (Figura A.7).

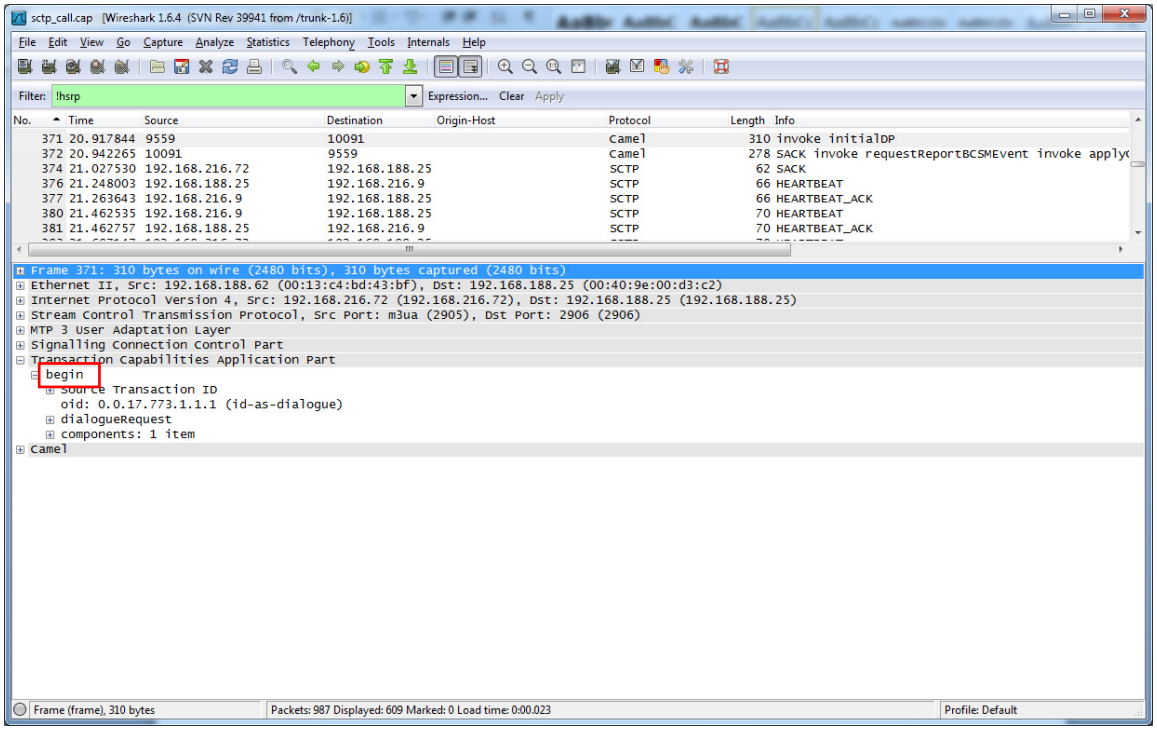


Figura A.7. Paquete TCAP

Una vez desempaquetada la información de TCAP, se obtiene los mensajes CAMEL que, como se puede observar en la Figura A.8, se trata de un IDP, es decir, de una solicitud de autorización de la MSC para poder conectar la llamada del abonado #A.

La información de la cual esta compuesto el IDP es:

- *Calling Party Number:* es el número telefónico del abonado que realiza la llamada, en este caso 593999563767.
- *CallingPartyNumber:* es igual al campo anterior pero decodificado de manera inversa de a par de dígitos, este caso amerita el ejercicio:
    - *CallingPartyNumber*: 0415959399657376.

    - Ordenado en pareja de dígitos: 04 15 95 93 99 65 73 76.
    - Ordenado inversamente: 40 51 59 39 99 56 37 67.
    - Se obtiene el prefijo: 4051 con el número *CallingPartyNumber:* 593999563767.
- *Location Number:* es el GT de la entidad que realiza la llamada, en este caso la MSC, 59397999126.
- *LocationNumber:* igual que el *LocationNumber* con la particularidad que está codificado de manera inversa por pareja de dígitos: 8497959397992106, dando como resultado el prefijo: 4879 el GT de la MSC: 59397999126 Y el sufijo: 0.
- *International Mobile Subscriber Identity* (*IMSI*): es el identificador de la línea del #A en el HLR, codificada de manera inversa por pareja de dígitos: 47000121820430f2, es decir que el IMSI: 740010122840032f, donde la “f” indica el fin de la cadena.

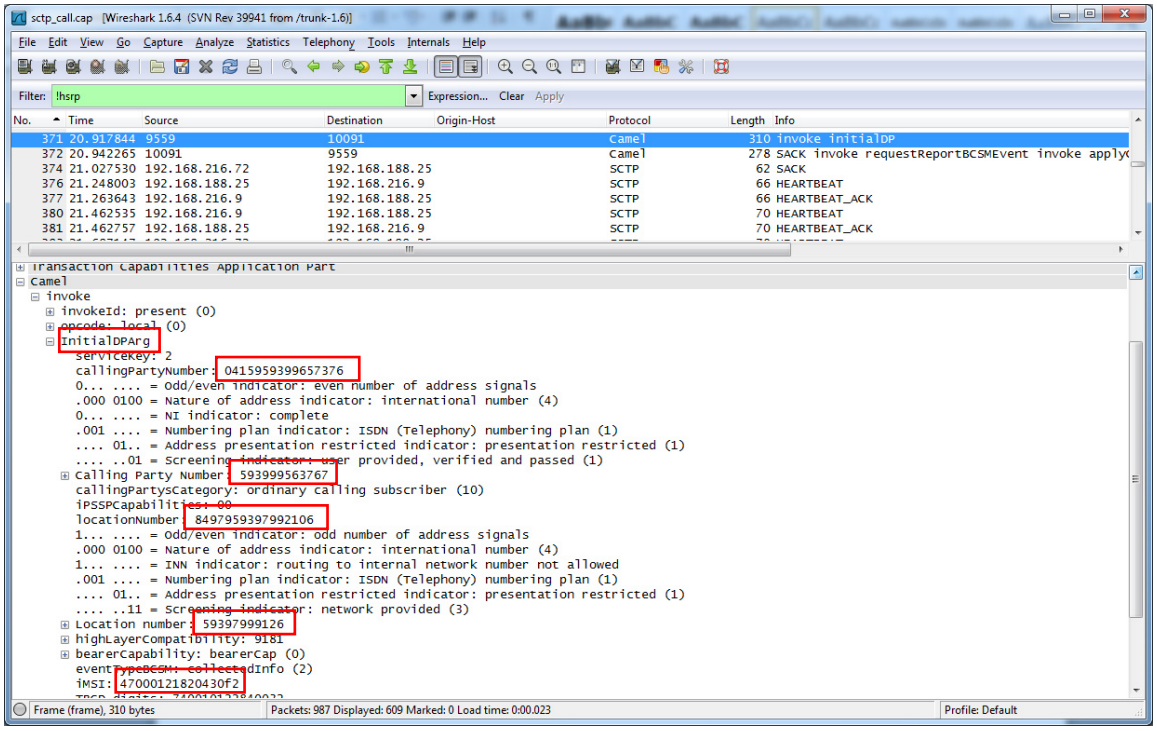


Figura A.8. Argumentos del mensaje IDP en CAMEL

Con la información obtenida en el IDP, la *Red Inteligente*, consulta en su base de datos la cuenta del abonado que pretende realizar la llamada, de tal manera que puede responder satisfactoriamente o no a esta solicitud, para el caso analizado se puede observar que la plataforma Prepago responde con los mensajes *Request Report BCSM Event*, *Apply Charging* y *Connect* (Figura A.9), autorizando a la MSC que proceda a conectar la llamada.

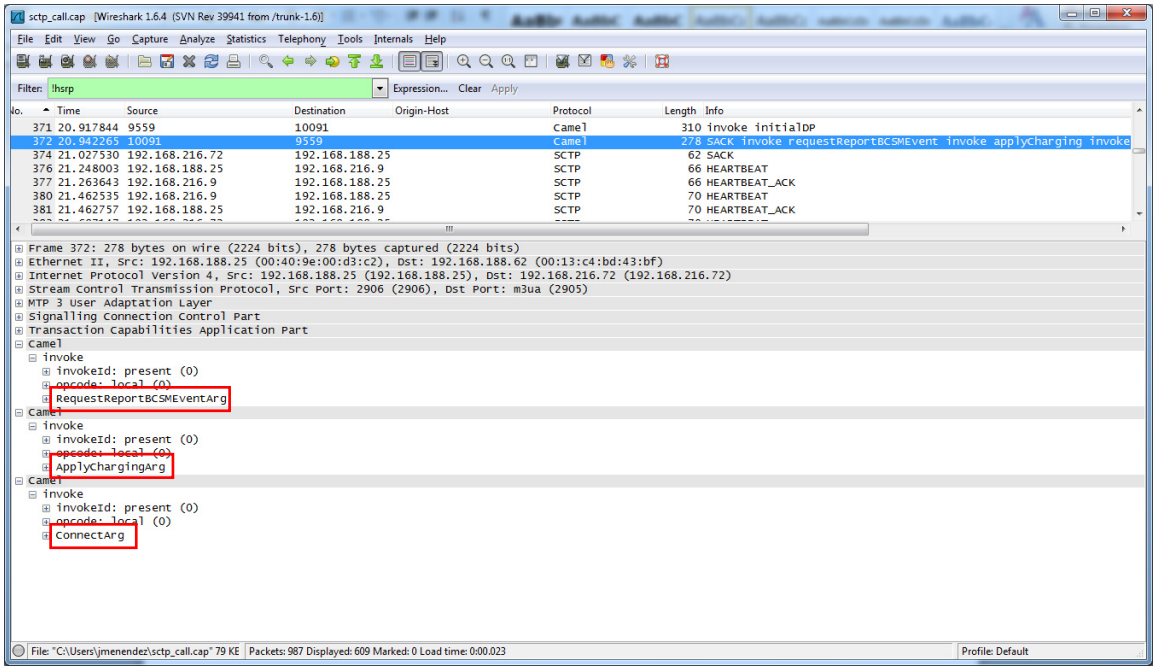


Figura A.9. Respuesta de la Red Inteligente a la MSC